# Istio at Scale: How eBay is building a massive Multitenant Service Mesh using Istio

Sudheendra Murthy

# Agenda

- Introduction
- Applications Deployment
- Service Mesh Journey
- Scale Testing
- Future Direction



# Introduction: eBay at a glance

| | |
|---|---|
| **185M** | **Number of Active Buyers worldwide** |
| **19M** | **Number of Sellers worldwide** |
| **1.7B** | **Number of Live Listings** |
| **$26.6B** | **GMV in Q4 2020** |

# eBay Applications

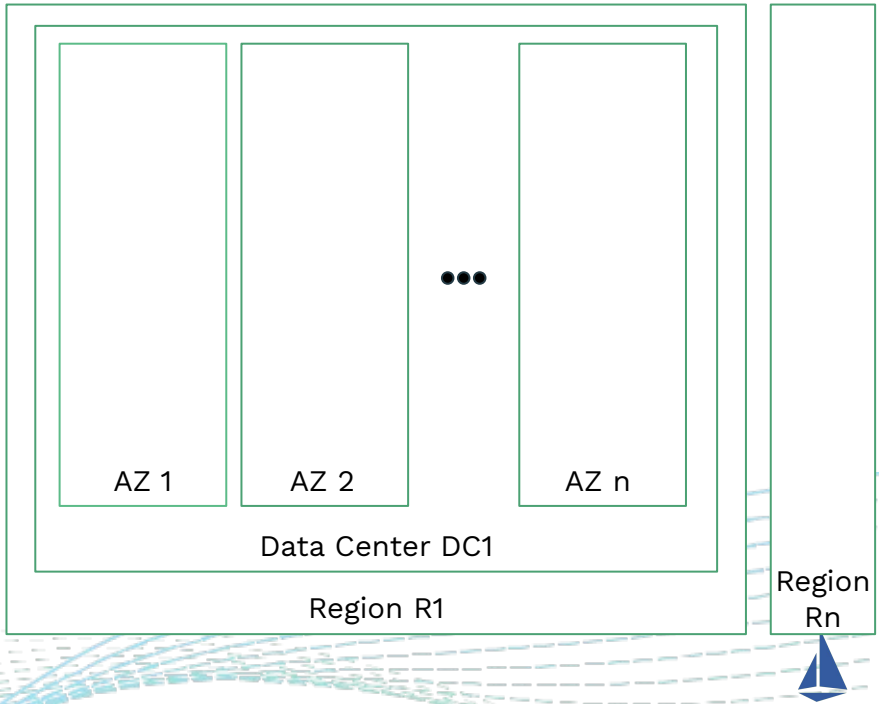
eBay is powered by

- More than 5,000 Microservices ranging from
  - API services, Search Engine, etc.
  - Databases, Key-Value stores - Oracle, MySQL, etc.
  - Big data systems & Pipelines - Hadoop, Apache Spark, Apache Flink, etc.
  - Machine Learning Platforms - Tensorflow, PyTorch, Jupyter Notebook, etc.
  - Central Logging & Tracing - Prometheus, ClickHouse, etc.
  - Messaging systems - Kafka, RabbitMQ, etc.
  - Programming Languages - Java, Python, Go lang, Scala, etc.
- Running on variety of Hardware
  - General-purpose x86 servers
  - GPUs



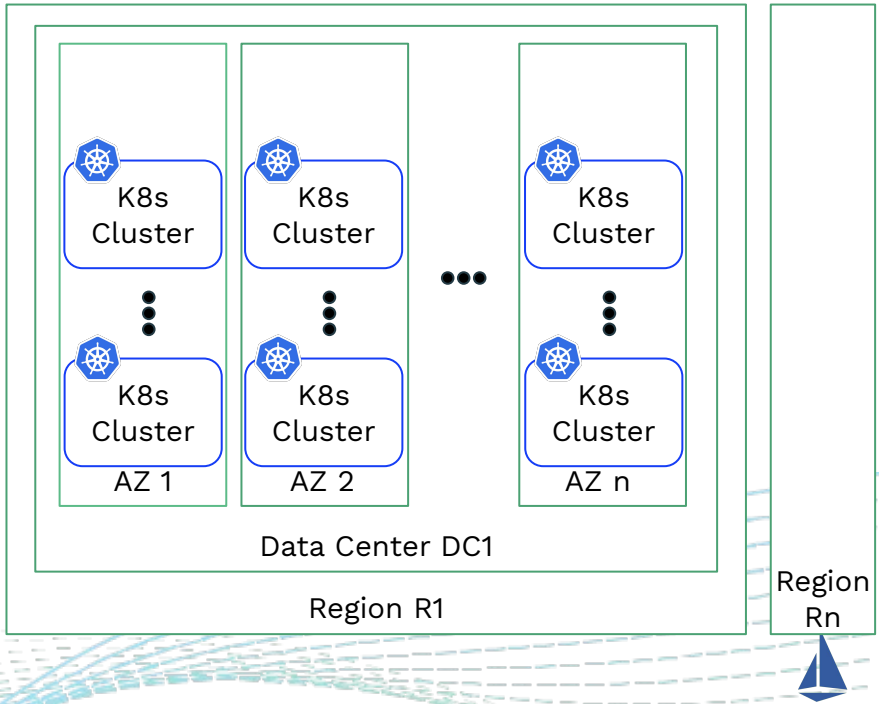
# Application Deployment: Cloud Layout

- **Region:** A metro region
- **DC:** One or more Data Centers in each Region
- **AZ:** One or more Availability Zones in each DC
  - Independent power, cooling, networking, etc.
- **PoP:** 20+ Points of Presence, locations across globe peering with the Internet closer to the customer
  - PoPs are mini AZs



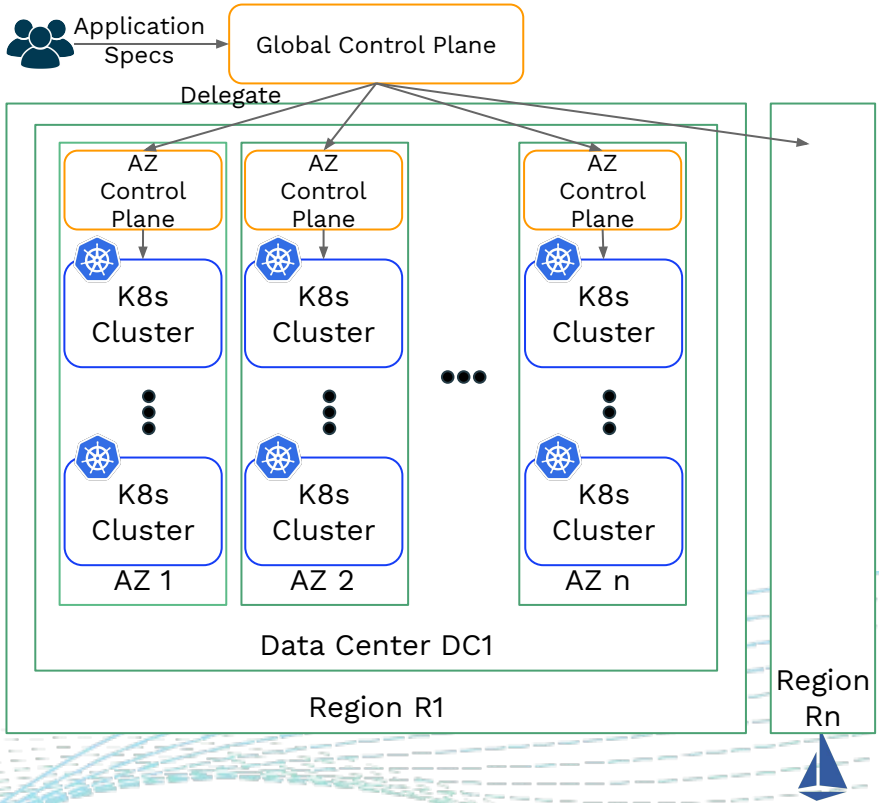
# Application Deployment: Cloud Layout

- **Multiple K8s Clusters** in an AZ
  - Each K8s cluster ~ 200 - 5,000 nodes
  - Upto 100,000 Pods in a cluster
  - 10,000+ K8s services - including prod, pre-prod, staging, etc.
- Applications deployment for HA
  - In all regions
  - In multiple AZs in each region
  - Capability to run all applications from a single region or AZ in a worst-case scenario



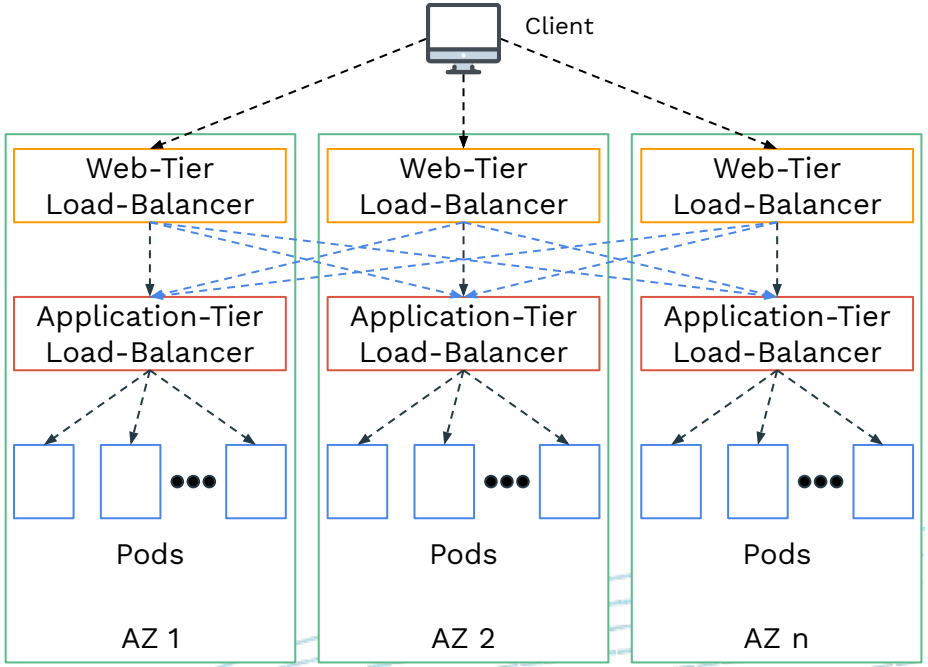
# Application Deployment: Federation

- Hierarchy of control planes
- Global Control Plane
  - Users provide application specs to Global Control-Plane
  - Syncs specs to AZ control-planes
  - Hosts global services - Global IPAM, Access-control Policy store, etc.
- AZ Control Plane
  - Syncs specs to workload K8s clusters in the AZ
  - Shared-Nothing Architecture
    - Hosts services catering to the AZ, e.g., AZ IPAM, Network Load-balancers, etc.
    - Full isolation by confining service failures to AZ boundary



# Load balancing & Traffic Flow

- Two tiers of hardware Load-Balancers (LB)
- Application-Tier LB
  - K8s service realized on Application-Tier LBs
- Web-Tier LB to control -
  - Percentage of traffic sent to an AZ, region, etc.
  - L7 routing
  - Hardware Firewalls (not shown) in front of Application-Tier LBs
- Client connects to closest Web-Tier LB based on DNS lookup



# What about Security?

- L4 Micro-segmentation Solution
  - Central Policy store capturing Application-to-Application dependencies
  - Controllers watch K8s clusters and translate policies into K8s NetworkPolicies to be enforced in the clusters
  - There are also other enforcers to enforce L4 policies on -
    - hardware Firewalls, Bare Metals, legacy OpenStack, etc.
- Transport Layer Security (TLS)
- Custom OpenID implementation for L7 AuthN

# Why Service Mesh?

- Current challenges include -
  - Manageability of Hardware Devices
    - Traffic Management & Security Enforcement
    - Updating hardware devices is slow
  - Achieving micro-segmentation at scale
  - Enabling TLS for all applications in a consistent way
- Service Mesh
  - An architectural pattern to implement common Security, Observability, Service Routing & Discovery functions as features of the infrastructure -
  - Functions: TLS Termination, Traffic Management, Tracing, Rate Limiting, Protocol Adapter, Circuit breaker, Caching, etc.



# Service Mesh Journey

## Step 1

**Declarative Intent**

- Capture application traffic characteristics as specs
- Realize Specs on Hardware LB

## Step 2

**Replace Hardware**

- Replace Hardware LBs & Firewalls

Current Status

## Step 3

**AZ Architecture**

- Evolve into AZ based architecture

## Step 4

**Evolving Security**

- Dial-tone security with Trust Domain
- L7 policy enforcement

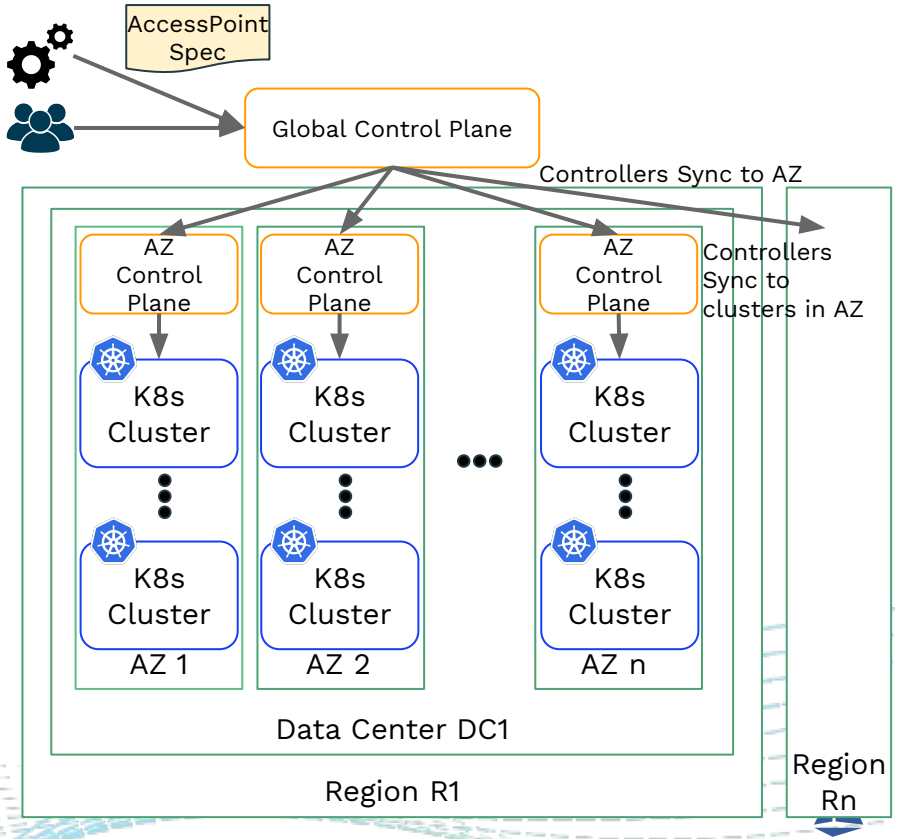
# Step 1: Access Point Spec

- Capture Traffic Management & Routing intent as *“Access Point”* Specs
  - Leverage Istio object model: Gateway, VirtualService, DestinationRules, etc.

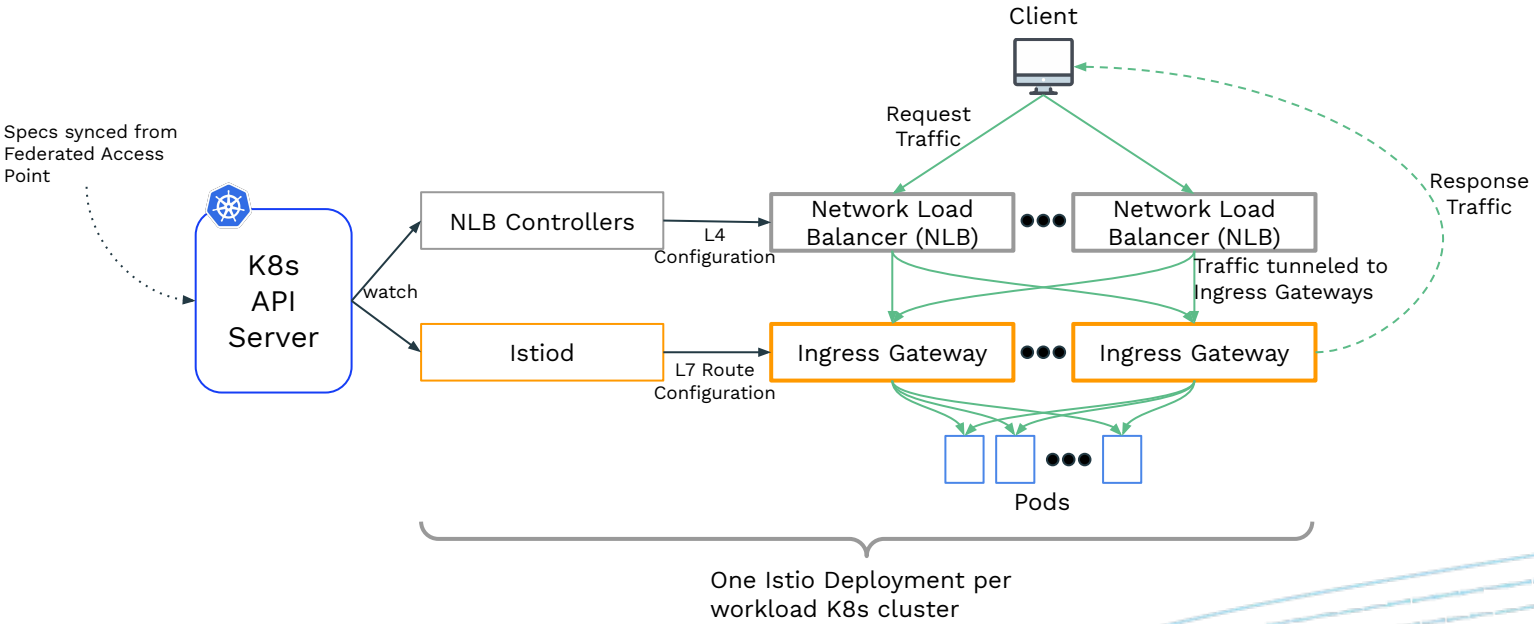
```
apiVersion: apps.cloud.io/v1
kind: AccessPoint
metadata:
  name: my-accesspoint
spec:
  accessPoints:
  - name: web-tier
    scopeIDs:
    - az1
    scopeType: AvailabilityZone
    traffic:
      gateways:
      - apiVersion: networking.istio.io/v1beta1
        kind: Gateway
        spec:
          ...
      virtualServices:
      - apiVersion: networking.istio.io/v1beta1
        kind: VirtualService
        spec:
          ...
      destinationRules:
      - apiVersion: networking.istio.io/v1beta1
        kind: DestinationRule
        spec:
          ...
      ...
      serviceEntries:
      - apiVersion: networking.istio.io/v1beta1
        kind: ServiceEntry
        spec:
          ...
      workloadEntries:
      - apiVersion: networking.istio.io/v1beta1
        kind: WorkloadEntry
        ...
  - name: app-tier
    scopeIDs:
    - cluster1
    scopeType: Cluster
    traffic:
      services:
      - apiVersion: v1
        kind: Service
        spec:
          ...
```

# Step 1: Access Point Spec

- Create the Specs on our Global Control Plane
- Realized on hardware LBs
- Internal orchestration & UI tools to use *Access Point* specs
- Standardization provides flexibility to switch backend implementations to software

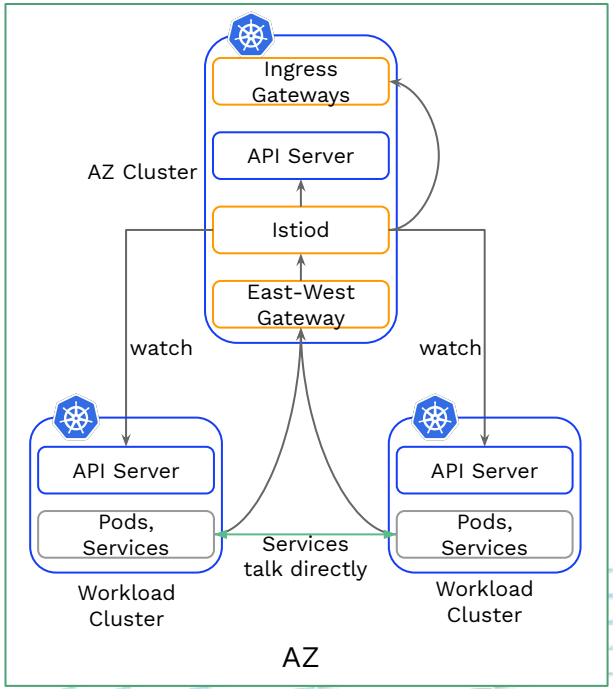


# Step 2: Replace Hardware LBs with Software



# Step 3: Evolve into AZ architecture

- One Istio deployment per K8s cluster is simple, but traffic between clusters in same AZ transits through Gateways
- To have service mesh span all clusters in an AZ -
  - Re-deployed Istio to AZ cluster
  - In *Primary-Remote* configuration within an AZ



# Step 4: Evolving Security

- Dial-tone security (Peer AuthN) using mutual TLS
  - Leverage SPIFFE Trust Domain
    - Trust Domain: Trust root of the system having separate root CA
    - Each workload gets unique identity based on K8s Service account -

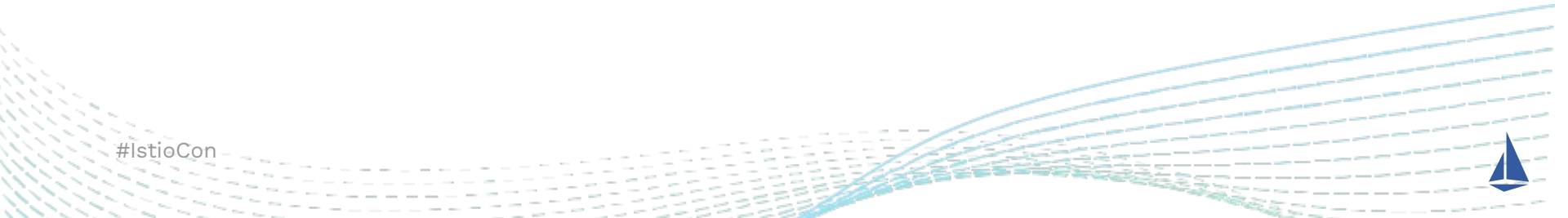
      *spiffe://<trust domain>/ns/<namespace>/sa/<service account>*
    - Following assertions enforced through admission checks -
      - Each namespace is globally unique across all clusters
      - Each deployment is associated with a unique service account
  - Trust Domain mapped to workload environments
    - Prod, Pre-prod, PCI, Staging, etc.
  - To support multiple trust domains in a single K8s cluster
    - Deploy multiple Istio deployments within a K8s cluster
    - Each Istio deployment manages subset of namespaces using DiscoverySelectors
  - Overall, create macro-segments for different environments



# Step 4: Evolving Security

- Origin or Request Authentication
  - Internal OpenID implementation for origin authentication
  - Plan to integrate with Istio





# How does it all scale ...?

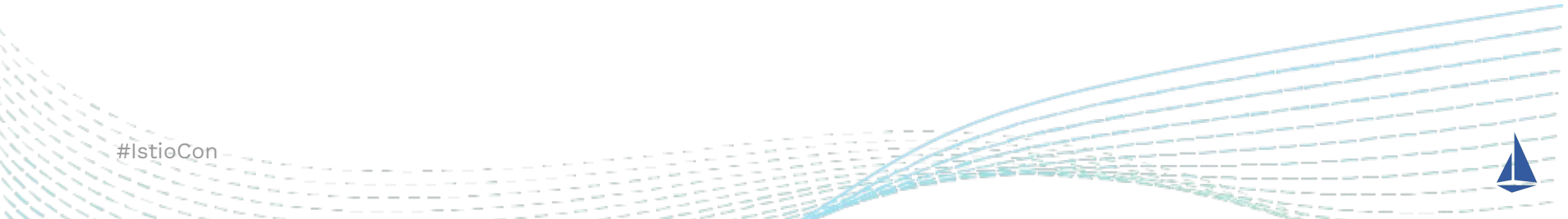
- Extensive Data-plane & Control-plane scale testing
- Data-plane performance of Envoy is well documented
- Control-plane scale testing
  - Primary Goal
    - Understand Istio control-plane performance to support eBay scale
    - Proxy config convergence time (CDS, EDS, LDS, RDS push times)
    - Resource usage (CPU, memory, etc.)
  - Secondary Goal
    - Fine-tune configuration params - debounce interval, push concurrency, etc.



# Control-plane Scale Testing: Setup

- Setup
  - Create Gateway Pods & thousands of Pods with sidecar Envoys
  - Measure Config convergence time
    - Time taken by *all* sidecars to get config from Pilot without any errors
    - For thousands of services & endpoints
    - With different churn rates of Pods





# Control-plane Scale Testing: Results

- Default wide-open egress sidecar configuration does not scale
  - Results in high memory usage & convergence times since each sidecar knows about all services in the cluster
  - Disabled egress traffic to restrict config pushed to sidecars
- Main Takeaways
  - P99.9 time from single Pilot instance to 0 - 3,000 sidecars **< 1 second**
  - Pilot CPU & memory within acceptable limits: **< 10 cores, 25 GB memory**
  - Pilot can scale horizontally
- Need to tune *PILOT_DEBOUNCE_AFTER*, *PILOT_DEBOUNCE_MAX*, *PILOT_PUSH_THROTTLE*, etc. params of Istio Pilot

# Future Direction

- Support for on-demand config pushes to Envoy via Incremental XDS
- Support for multiple trust domains & namespace isolation natively in Istio
- Bridging trust between gateways of different AZs
  - Mutual TLS between Pods of same environment across AZs
- Scaling Authorization Policies
  - Millions of policies
  - Global Identity federation

# Thank you!

Contact us:

DL-eBay-ServiceMesh@ebay.com

https://www.linkedin.com/in/sudhimurthy/

#IstioCon