Are Opensource Cloud Technologies Ready for Enterprise Scale?

Surya V Duggirala

IBM
Open Source Software Everywhere
Speaker Profile

IBM Cloud Engineering Guild Leader

Co-chair, Istio Performance Workgroup

Global Technical Ambassador (GTA)

Surya V Duggirala
Open source allows developers to make a difference
Fueling most of the cloud platforms in the industry
Cloud Scale Stats from Multiple Industry Domains

- **Weather**
  - Support 211M Video starts, 87k concurrent live stream views during a severe hurricane

- **Healthcare**
  - Support about 7 Million customers + Handling Millions of Immunization records for schools

- **Banking**
  - Retail Banking solutions with more than 6 Million customers + 50K Commercial Banking Customers

- **Airlines**
  - Support 6700 flights per day to over 350 destinations with web, mobile and kiosk to Cloud

- **Car Rental**
  - Support 500 Million total DB transactions per day

- **Consumer Appliances**
  - Support more than 2 Million Appliances with IoT API latency less than 500 ms
What are we going to talk in Today’s Session?

- Opensource Frameworks in Cloud
- Kubernetes Safe Scheduler (SSX)
- Istio OSS Focus on Scale
- Other Open Community Initiatives
- Summary
Cloud Platform Architecture – Opensource Technologies

- **Content**
  - Helm

- **Service Mesh**
  - Istio

- **Platform Services**
  - Prometheus
  - Fluentd
  - Jaeger
  - OpenTracing
  - Elastic Stack
  - Grafana

- **Orchestration**
  - K8s

- **Container Runtimes**
  - Docker
  - rkt
  - CRI-O
  - kata containers
  - Firecracker

- **Virtualization**
  - KVM
  - vmware ESXi
  - ZVM

- **Operating System**
  - Linux
  - OpenSUSE
  - container linux
  - Zowe

- **Hardware**
  - Intel
  - OpenPOWER

**Applications, Middleware, Services for Cloud**

**Behavioral Insights and Operational Control for microservices**

**Standardized Monitoring, Logging and Tracing for Cloud**

**De facto Operating System for Cloud**

**Lightweight Containers and Micro-VMs**

**Virtualization for Cloud**

**Container Friendly Operating Systems**

**Hardware Exploitation with Open Cloud Technologies**
Persistent Memory CSI Plug-in

Device Plugins for Acceleration to Security and Compression

CPU Manager for Kubernetes

Node Tagging for Hardware Profiles

GPU Plugin for Kubernetes

- K8s API Server Scalability #1 – Exploit Hardware Acceleration
- Container Density #2 – Exploit low latency Persistence Memory
- Noisy Neighbor Problem #3 - Constraining Workloads to CPUs
- Hardware Centric Feature Identification #4 – Node Tagging

https://01.org/kubernetes/building-blocks
Virtualization

- Common Hypervisor is Xen Hypervisor on bare metal
- Common types of Virtualization on Xen are HVM (Hardware Virtual Machine) and PV (Paravirtualization)
- Virtual Machines run on top of Hypervisors
- Hypervisors take care of CPU Scheduling and Memory partitioning
- Hypervisor is unaware of Networking, external storage devices or common I/O functions

Differences between various Virtualization Types (PV vs. HVM)
- HVM Guests will have less Kernel CPU Usage
- HVM also better for storage and network I/O
- Final choice is based on application usage needs
Virtualization Technology Impact

- Kernel CPU on HVM is only 1/3 that of PV VSI node saving significant CPU cycles
- For services like Cloud Foundry, HVM gives almost 7x more density packing almost 200 containers per cell
- Containerd has density advantages to Docker
- Kata Containers and micro-vms are gaining ground
- Nested Container architectures with few clouds
- Sidecars and Pod density considerations
- CPU sharing algorithms and noisy neighbor considerations

<table>
<thead>
<tr>
<th>Test</th>
<th>Docker</th>
<th>Containerd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of pods running without node occasionally going NotReady</td>
<td>400</td>
<td>475</td>
</tr>
<tr>
<td>Time for 400 pods to become ready (100 at a time)</td>
<td>11 mins</td>
<td>3 mins</td>
</tr>
</tbody>
</table>

Orchestration

- Default K8s Scheduler algorithm may result in unbalanced clusters
- Default K8s Scheduler depends on static request values
- Descheduler has many Issues
- What we need is a Rescheduler with dynamic cluster insights

- Smart Behavior through Extension
- Enhance Default Scheduler with Smart/Safe Scheduler
- Combination of K8s Extension and Node Annotator
- Support Over-commitment
Safe K8s Scheduler (SSX) Design

- SSX extends the predicates and priority functions of the default scheduler
- It supports Kubernetes 1.13.5
- Two priority functions are provided: 'safe-overload' and 'safe-balance'
  - The former orders nodes based on their calculated risk values
  - The latter orders nodes according to a combined measure of average and standard deviation of the available resources
  - Such a combined measure allows a trade-off between the average load and the variability in the load

[https://github.com/IBM/kube-safe-scheduler](https://github.com/IBM/kube-safe-scheduler)
SSX Scheduler - Design

Overview

- Data analyzer
  - Profiling
  - Characterization
  - Prediction

- Scheduler
  - Optimizer
    - Node
    - Cluster

- Monitor

Nodes

- Resource usage data

Usage model parameters

Container data

Node aggregated data

Data modeling

Effective availability

Optimization
Over-subscription policy

- Set parameters
  - Load threshold ($U^*$)
  - Overload probability ($\epsilon$)
- Explore Mean-Variance (MV) space
- Select operating point
  - Low load: afford variability
  - Medium load: calculated variability
  - High load: limited variability
- Results in an effective resource availability metric to be used in a scheduling predicate
SSX Scheduler - Configuration

SSX may be configurable through the following environment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAFEUTILIZATION</td>
<td>integer to set the threshold utilization value</td>
</tr>
<tr>
<td>SAFEPERCENTILE</td>
<td>integer to set the percentile acceptable value</td>
</tr>
<tr>
<td>SAFEFORECASTWEIGHT</td>
<td>integer to set weight percentage of forecast value</td>
</tr>
<tr>
<td>SAFEPRINTTABLE</td>
<td>boolean to select detailed logging information</td>
</tr>
</tbody>
</table>

An example follows.

```yaml
env:
  - name: SAFEUTILIZATION
    value: "90"
  - name: SAFEPERCENTILE
    value: "30"
  - name: SAFEFORECASTWEIGHT
    value: "20"
  - name: SAFEPRINTTABLE
    value: "false"
```

https://github.com/IBM/kube-safe-scheduler
- Monitoring Service Scalability #1 – Prometheus operator with tuning knobs like scraping interval can help scale
- Tracing Scalability #2 – Jaeger operator with sampling can help
- Projects like Thanos can help provide global query view and merge data from Prometheus HA pairs with massive storage
- Exploiting faster persistent memory systems from latest processors can significantly help scale monitoring systems on cloud

- Application Developer Productivity #1 – Significant improvements to application developer productivity
- Observability #2 – Best way to control large microservice meshes
- Managed Service Mesh #3 - Cloud Providers are providing managed service mesh taking out complexities and scale issues
- Self Consumption by Cloud Providers#4 – Many of the internal frameworks of cloud and many cloud services started using Service mesh
Istio Service Mesh – What is It?

Connect

Secure

Control

Observe

Intelligent Routing
Load Balancing

Service to Service
Authentication
Certificate Management

Policy Enforcement
Authorization

Telemetry, Logging,
Visualization
Distributed Tracing

https://istio.io/
Istio Open Source Project – Architecting for Scale

- **Istio**
  - SWAT Team
    https://docs.google.com/document/d/1TiMmzbM5r9QlBfs79owBqiAZJgfLMyXa3Zuev7xZKSc/edit#
  - SWAT Team Report
    https://docs.google.com/document/d/1ob9M2TfI5GS3_rqrEhWFIpVyfFVKPTE8N3YZpeiDJxw/edit#
  - Istio Performance and Scalability Optimization Issues on GH
    https://docs.google.com/document/d/17PfZkXGpiVk3qAQdnB_ASSIGvnV7ZKwH6RXBN3c8_so/edit
  
- **Istio Community Regression Patrol Framework**
  - Istio Code Quality https://ibmcloud-perf.istio.io/regpatrol/
<table>
<thead>
<tr>
<th>#</th>
<th>Focus Area</th>
<th>Feature</th>
<th>Performance Benefits</th>
<th>Feature Design Details</th>
</tr>
</thead>
</table>
| 1  | Telemetry    | • Sampling • New Control points | Significant reduction in Istio Control plane resource usage    | • Sampling feature provides the ability to collect a specific sample set of telemetry data instead of collecting every metric  
• With new control features, end users will have ability to control metric collection either at the source or destination |
| 2  | Pilot        | Dynamic load distribution     | Uniform traffic distribution from proxies to Pilot replicas in the mesh | Grpc-lib centric more dynamic load balancing of traffic between Envoys and Pilot replicas |
| 3  | Proxy Sidecars | Improve network Bandwidth & resource reduction | • Reduce the container network costs by bypassing kernel with direct app to envoy through user space  
• Save memory copies and fragmentation  
• Set the correct worker threads | • Using eBPF save almost 60% BW costs  
• Need to set the correct Envoy worker thread count for multiple host environments either statically or dynamically |
| 4  | Buffer and Cache management | Externalize Buffer and Cache tuning at data plane | Improve the agility and density at data plane                  | With this feature end users have the ability to configure buffer and cache at data plane and control plane level |
| 5  | Envoy        | Further perf improvements      | Perf benefits to both side cars and Gateways                   | Currently with HTTP1.1 traffic, Envoy performs much slower than HAProxy (need to fix this) |
## Istio Performance Future Design Optimizations for Scale

<table>
<thead>
<tr>
<th>#</th>
<th>Focus Area</th>
<th>Feature</th>
<th>Performance Benefits</th>
<th>Feature Design Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>InitContainer</td>
<td>Cold Start</td>
<td>Required for Knative to reduce cold startup time</td>
<td>• Any init container adds significant cold-start time due to paying kubelet startup cost</td>
</tr>
<tr>
<td>7</td>
<td>Istio Proxy</td>
<td>Startup time</td>
<td>Required for Knative to reduce istio proxy startup time</td>
<td>We see a base of ~200ms cost of pushing envoy config down from pilot and more at scale. This is a significant portion of our cold start budget so we need a way to optimize this to near 0. Statically defining the northbound and other endpoints which our nodes can reach and not participating in the actual 'mesh' is an option</td>
</tr>
</tbody>
</table>
Standardized packaging solutions through open source mechanisms
Cloud Packs through Helm and Operators is becoming a standard way to deliver Content and services
Operators provide the ability to automate common operational tasks through scripted controllers

Frameworks Integration - Opensource Cloud Technologies

OSS Frameworks Integration

- Cloud Foundry and Istio Integration
  - https://docs.google.com/document/d/1LgLY0g39fzpg1_4zTckbH1mOuuSKGvYwp2tkakoe9ys/edit#heading=h.tj1oxhc47cj

- Cloud Foundry and Kubernetes Integration
Key Takeaways from this Session

- **Open Source Cloud Technologies** are vetted for enterprise scale
- **SSX K8s Scheduler** will help enhance default K8s scheduler algorithm to safely balance clusters with over-commit
- **Istio Performance Workgroup** introduced a regression patrol framework to maintain code quality on a daily basis
- **Hardware Exploitation** will help reduce TCO for Cloud
- **Operating Systems** are getting optimized for containers
- **Operators** are becoming the common delivery mechanism for content for production use
- **Virtualization** technologies selection will impact container density
Thank You

Q&A

suryadu@us.ibm.com
https://www.linkedin.com/in/suryaduggirala1
@Duggirala1