Real-Time Analytics Meets Kubernetes

Tal Doron
Director, Technology Innovation
ABOUT ME

@taldoron
taldoron84
tald@gigaspaces.com

Tal Doron
Director, Technology Innovation
About GigaSpaces

We provide one of the leading in-memory computing platforms for real-time insight to action and extreme transactional processing. With GigaSpaces, enterprises can operationalize machine learning and transactional processing to gain real-time insights on their data and act upon them in the moment.

InsightEdge is an in-memory real-time analytics platform for instant insights to action; analyzing data as it's born, enriching it with historical context, for smarter, faster decisions

In-Memory Computing Platform for microsecond scale transactional processing, data scalability, and powerful event-driven workflows

300+
Direct customers

50+/500+
Fortune / Organizations

5,000+
Large installations in production (OEM)

25+
ISVs
Why

* Intro pictures from Wikipedia
We’ve looked up to the stars
Not without first passing through the clouds
It’s the smallest of opponents that are gamechangers.
We needed to find a way to ship man there...

The first flight of an airplane, the Wright Flyer on December 17, 1903
How do we become cloud native?

- **Manage Large Deployments**
  - Cloud-ready, ZooKeeper based for large-scale and federated deployments
- **REST API Management**
  - Standards-based, utilizing
- **Containerization and Orchestration**
  - Docker, Kubernetes, OpenShift etc.
- **Application-driven Deployment**
  - Serverless-like user experience
- **Pluggable Elastic Resource Balancing**
  - Scheduling for dynamic re-partitioning and resource allocation
- **Telemetry and Cluster Intelligence**
  - Predictive maintenance / fault-tolerance over large-scale deployments
Who’s using K8s?
OVERVIEW

• An overview of Kubernetes and the value it is bringing for automating deployment, scaling, and management of containerized applications

• How organizations can simplify management and container deployment on Cloud, Hybrid or On-premises environments with GigaSpaces InsightEdge

• 3 top open-source tools for production: HELM, Istio, and Prometheus

• A Kubernetes services comparison between cloud providers: AWS vs. Azure vs. GCP
How Can You Gain the Most Value from Your Data?

Near real-time data is highly valuable if you act on it on time.

Historical + near real-time data is more valuable if you have the means to combine them.

- Near real-time data is highly valuable if you act on it on time.
- Historical + near real-time data is more valuable if you have the means to combine them.
InsightEdge: Real-time Analytics for Instant Insights To Action

**VARIOUS DATA SOURCES**

- TensorFlow
- Spark
- SQL
- Desktop Applications

**REAL-TIME LAYER**

- RAM
- STORAGE-CLASS MEMORY
- SSD STORAGE

**DISTRIBUTED IN-MEMORY MULTI MODEL STORE**

**REAL-TIME LAYER**

- HOT DATA
- WARM DATA

**BATCH LAYER**

- COLD DATA

**APPLICATION**

- REAL-TIME INSIGHT TO ACTION

**DASHBOARDS**

- No ETL, reduced complexity
- Built-in integration with external Hadoop/Data Lakes S3-like
- Fast access to historical data
- Automated life-cycle management
At least 54% of the Fortune 500 were hiring for Kubernetes skills in 2017.

Around 51% growth for Kubernetes share in the market in 2018.
Kubernetes is the Winner

- #1 discussed project on GitHub
- Top 2 in number of contributors
- ~400K users on Slack

Which distribution of Kubernetes are you using?
- Vanilla Kubernetes
- OpenShift
- Rancher
- CoreOS Tectonic
- Heptio
- Google Kubernetes Engine (GKE)
- Azure AKS
- Other
Business Landscape

• The leading orchestration tool vs. Docker Swarm, Mesos, OpenShift and Cloud Foundry and most used CNCF project

• All cloud vendors have a managed Kubernetes service (EKS, AKS and GKE)

• Apache Spark 2.3 has native Kubernetes support
Why Kubernetes?

Key building blocks for a “cloud like” platform as a service

- **Auto deployment** of data services, functions and frameworks (Spark ML, SQL, Zeppelin, etc.)
- **Orchestration automation** with cloud native solutions (auto scale, self healing)
Kubernetes – Management POD

- **Lookup Service (LUS)** - The Lookup Service provides a mechanism for services to discover each other. For example, querying the LUS to find active GSCs.
- **Apache ZooKeeper** - Zookeeper is a centralized service used for space leader election.
- **REST Manager** - RESTful API for managing the environment remotely from any platform.
**Kubernetes – Data POD**

- **Data Grid Instance** – This is the fundamental unit of deployment in the data grid. A Processing Unit instance is the actual runtime entity.
- Each Data POD contains a single instance to provide cloud native support using Kubernetes built-in controllers (auto scale, self healing)
• **Driver Pod** – The Spark driver is running within a POD. The driver creates executors, connects to them, and executes the applicative code.

• **Executor Pod** – When the application completes, the executors’ pods terminate and are cleaned up, but the master pod persists logs and remains in “completed” state.
XAP High Level Overview 3.1
Kubernetes Dashboard View

CPU usage

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Restarts</th>
<th>Age</th>
<th>CPU (cores)</th>
<th>Memory (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Running</td>
<td>0</td>
<td>12 hours</td>
<td>0.005</td>
<td>269.184 Mi</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>0</td>
<td>12 hours</td>
<td>0.004</td>
<td>354.195 Mi</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>0</td>
<td>12 hours</td>
<td>0.005</td>
<td>286.137 Mi</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>0</td>
<td>12 hours</td>
<td>0.004</td>
<td>277.977 Mi</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>0</td>
<td>12 hours</td>
<td>0.005</td>
<td>279.773 Mi</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>0</td>
<td>12 hours</td>
<td>0.005</td>
<td>275.051 Mi</td>
</tr>
</tbody>
</table>
“Under the Hood” Guidelines

• Apply a **POD Anti-Affinity** using label selectors for both Data and Management PODs
  • For example: spread the primary and backup data pods from this service across zones

• Each POD has a persistent identifier that is maintained across any rescheduling using **StatefulSets**
  • For example: automated rolling updates/scale up data pod one-by-one
**Installation**

- HELM – The package manager for Kubernetes
- Helm Charts helps you define, install and upgrade both XAP and InsightEdge

```bash
# helm install gigaspaces/insightedge --version=14.0 --name demo
```
Installation – Define Capacity

- The following Helm deploys a cluster with 3 partitions with 512MiB allocated for each partition:

```
# helm install gigaspaces/insightedge --version=14.0 --name demo
--set pu.partitions=3,pu.resources.limits.memory=512Mi
```
The following Helm command deploys a cluster in a high availability topology, with anti-affinity enabled:

```bash
# helm install gigaspaces/insightedge --version=14.0 --name demo --set pu.ha=true,pu.antiAffinity.enabled=true
```
Testing for Liveness

- Use liveness probes to notify Kubernetes that your application’s processes are unhealthy and it should restart them
- The probe calls a bash script

```yaml
livenessProbe:
  exec:
    command:
      - sh
      - -c - "data-pod-liveness 3181"
  initialDelaySeconds: 15
  timeoutSeconds: 5
```
Testing for Readiness

• Use **readiness probes** to notify Kubernetes that your application’s processes are able to process input, for example: when data is loading the pod not yet ready.

• The probe calls a bash script

```yaml
readinessProbe:
  exec:
    command:
      - sh
      - -c - "data-pod-ready 2251"
  initialDelaySeconds: 15
  timeoutSeconds: 5
```
WAN Gateway
1. Updates in New York cluster are pushed to local Delegator
2. Delegator sends the updates to the list of target sites configured in New York Gateway
3. London Sink will write the data to London Cluster
4. Any conflicts that occur are resolved using the custom Conflict Resolution algorithm
Auto Pod Failover
Auto Pod Failover

1. Data Pod B Fails
Auto Pod Failover

1. Data Pod B Fails
2. Failover to Data Pod B'
Auto Pod Failover

1. Data Pod B Fails
2. Failover to Data Pod B'
3. Data B is back up
Auto Pod Failback

1. Data Pod B Fails
2. Failover to Data Pod B’
3. Detect failure and restart Pod B
4. Once ready failback to Pod B as “proffered primary”
Automated Rolling Scale Up

1. Take Down Pod A’
2. Restart Pod A’ with X2 RAM
3. Fail over to Pod A’ and restart Pod A with X2 RAM
4. Fail back to Pod A

Repeat for each Pod
<table>
<thead>
<tr>
<th>Feature/Service</th>
<th>GCP</th>
<th>Azure</th>
<th>AWS</th>
<th>IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic Update</td>
<td>Auto or On-demand</td>
<td>On-demand</td>
<td>On-demand</td>
<td>On-Demand</td>
</tr>
<tr>
<td>Auto-scaling nodes</td>
<td>Yes</td>
<td>No, available thorough k8s autoscale</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Node Pools</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Multiple Zones</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RBAC</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Bare Metal Nodes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
3 Key Technologies for Kubernetes

- **Prometheus – Monitoring**
  Monitor applications and infrastructure running in Kubernetes, supports service discovery, built-in alerts, and more.

- **Istio – Service Mesh**
  Istio manages and routes encrypted network traffic, balances loads across microservices, enforces access policies, verifies service identity and provides tracing, aggregates service to service telemetry.

- **Helm – Package Manager for Continuous Deployments**
  Repeatable deployments without all of the overhead and complication of keeping dependencies up to date and consistent
RECORDED DEMO

LINK: https://www.youtube.com/watch?v=i4Z4l8N9Q
Fetch InsightEdge Helm Chart

List existing repos:
$ helm repo list

Add the GigaSpaces repo:
$ helm repo add gigaspaces https://resources.gigaspaces.com/helm-charts

Get updates:
$ helm repo update

Search GigaSpaces charts:
$ helm search gigaspaces

Fetch the InsightEdge 14.0 chart and untar it:
$ helm fetch gigaspaces/insightedge --version 14.0 --untar
Installing a Data Grid

Install a clustered data grid with two partitions, each with a high availability backup:

```
$ helm install insightedge --name demo --set pu.partitions=2,pu.ha=true
```
Monitoring

Using Helm

```bash
$ helm status demo
```

Using Kubernetes Command Line

```bash
$ kubectl get pods
```

Using Kubernetes Dashboard

```bash
$ minikube dashboard
```

Using GigaSpaces REST Manager

`http://192.168.99.100:30890`

Using GigaSpaces Command Line

```bash
$ insightedge --server=192.168.99.100:30890 space list
```
Running a Spark job

Run the following InsightEdge submit script for the SparkPi example. It calculates a Pi approximation. The result of the calculation is printed to the log.

```
work dir: gigaspaces-insightedge-enterprise/insightedge/bin
--name spark-pi --class org.apache.spark.examples.SparkPi --conf
spark.kubernetes.authenticate.driver.serviceAccountName=spark --conf
spark.kubernetes.container.image=gigaspaces/insightedge-enterprise:14.0
local:///opt/gigaspaces/insightedge/spark/examples/jars/spark-examples_2.11-2.3.2.jar
```

(Go to the driver pod and see the Pi value that was calculated, e.g. “Pi is roughly 3.1391756458782296”)

Running an InsightEdge Spark Job

Run the following InsightEdge submit script for the SaveRDD example, which generates 100,000 Products, converts them to RDD, and saves them to the data grid.

```
work dir: gigaspaces-insightedge-enterprise/insightedge/bin
```
Apache Zeppelin


## Interpreter Properties

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>default.driver</td>
<td>com.gigaspaces.jdbc.Driver</td>
</tr>
<tr>
<td>default.url</td>
<td>jdbc:insightedge:spaceName=demo?locators=demo-insightedge-manager-hs</td>
</tr>
</tbody>
</table>
SQL Queries

The following SQL Queries can be executed to analyze the data in the data grid.

```sql
%insightedge_jdbc
SELECT * from Product
SELECT count(*) from Product
SELECT id, quantity from Product where id<15
```

<table>
<thead>
<tr>
<th>description</th>
<th>featuredProduct</th>
<th>id</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description of product 99918</td>
<td>true</td>
<td>99918</td>
<td>7</td>
</tr>
<tr>
<td>Description of product 99596</td>
<td>false</td>
<td>99596</td>
<td>3</td>
</tr>
<tr>
<td>Description of product 99274</td>
<td>false</td>
<td>99274</td>
<td>9</td>
</tr>
<tr>
<td>Description of product 98998</td>
<td>true</td>
<td>98998</td>
<td>8</td>
</tr>
<tr>
<td>Description of product 98952</td>
<td>true</td>
<td>98952</td>
<td>9</td>
</tr>
<tr>
<td>Description of product 98906</td>
<td>false</td>
<td>98906</td>
<td>2</td>
</tr>
</tbody>
</table>
SQL Queries

```
SELECT count(*) from org.insightedge.examples.basic.Product
```

```
count(*)
```

```
100000
```

```
SELECT id, quantity from org.insightedge.examples.basic.Product where rownum<10
```

```
48960  48974  48998  49260  49488  49648  99090  99412  99734
```
Failover

1. Using command line, list Space instances and check which Space is elected primary

   work dir: gigaspaces-insightedge-enterprise/bin/

   ```
   $ insightedge --server=192.168.99.100:30890 space list-instances demo
   ```

   Example output:
   
<table>
<thead>
<tr>
<th>INSTANCE ID</th>
<th>MODE</th>
<th>HOST ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>demo~1_1</td>
<td>PRIMARY</td>
<td>demo-insightedge-space-1-0</td>
</tr>
<tr>
<td>demo~1_2</td>
<td>BACKUP</td>
<td>demo-insightedge-space-1-1</td>
</tr>
</tbody>
</table>

2. Using Kubernetes Dashboard, ‘Exec’ into the Pod of the primary Space instance
Failover

3. Execute the following command

   $ pkill -9 java

4. Execute list Space instances command line again and verify new elected primary

   Example output:
   
<table>
<thead>
<tr>
<th>INSTANCE ID</th>
<th>MODE</th>
<th>HOST ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>demo~1_1</td>
<td>BACKUP</td>
<td>demo-insightedge-space-1-0</td>
</tr>
<tr>
<td>demo~1_2</td>
<td>PRIMARY</td>
<td>demo-insightedge-space-1-1</td>
</tr>
</tbody>
</table>
To make a long story short, we’ve built spaceships.
THANK YOU

@taldoron

@taldoron84

tald@gigaspaces.com

Tal Doron
Director, Technology Innovation