Apache Ignite Service Grid
Backbone of Your Microservices-based Solution

Akmal Chaudhri
GridGain Systems

http://ignite.apache.org #apacheignite
Agenda

• Apache Ignite Service Grid
• Microservices-based Solution with Apache Ignite
  • Data Nodes
  • Service Nodes
  • Inter-communication
  • Internal and External Applications
  • Persistence
• Demo
Apache Ignite Service Grid
Apache Ignite Overview

Memory-Centric Storage
Scale to 1000s of Nodes & Store TBs of Data

Ignite Native Persistence
(Flash, SSD, Intel 3D XPoint)

Third-Party Persistence
Keep Your Own DB
(RDBMS, HDFS, NoSQL)
Apache Ignite Service Grid

• Any Service: counter, ID generator, etc.
• Cluster Singleton
• Node Singleton
• Load Balancing
• Fault Tolerant
Deployment and Load Balancing

- Manageable Deployment
  - Node Filter
- Service Requests
  - Service Proxy
  - Sticky vs Non sticky
- Manageable Deployment + Proxy
  - No need to have DDLs on all the nodes
  - Foundation for Microservices Architecture
Microservices-Based Solution With Apache Ignite
Microservices: Common Pitfalls

- Lifecycle Management
- Services Communication
- Load Balancing
- Fault-tolerance
- Scalability
  - Data Layer
  - Computational Layer
External Apps That Use Micro-Services

Ignite Cluster
Service Node
MS1
MS2
Service Node
MS1
Service Node
MS2
MS3
Data Node
Data Node
Data Node
Data Node

Persistent Storage

MongoDB
Cassandra
Hadoop
**Ignite Nodes**

- **Server Nodes**
  - Act as containers for data and computations
  - Generally started as standalone processes
- **Client Nodes**
  - Provide a cluster entry point to run operations
  - Embedded in applications code
Microservices Architecture: Data Node

• Server node that
  – Stores data
  – Run queries and computations

• Plain Distributed Storage
  – No data model classes
  – No computations classes
  – No services classes

• No need to restart
public class DataNodeFilter implements IgnitePredicate<ClusterNode> {
    public boolean apply(ClusterNode node) {
        Boolean dataNode = node.attribute("data.node");
        return dataNode != null && dataNode;
    }
}

<property name="userAttributes">
    <map key-type="java.lang.String" value-type="java.lang.Boolean">
        <entry key="data.node" value="true"/>
    </map>
</property>

<bean class="org.apache.ignite.configuration.CacheConfiguration">
    <property name="name" value="vehicles"/>

    <property name="nodeFilter">
        <bean class="common.filters.DataNodeFilter"/>
    </property>
</bean>
Microservices Architecture: Service Node

- Server or client node that
  - A candidate for a service deployment
  - Handles service requests
- Hosts one or many services
  - Same type instances
  - Service A, Service B, etc.
- Updated separately
  - New service version released
  - Restarted as a group
public class VehicleServiceFilter implements IgnitePredicate<ClusterNode> {

    public boolean apply(ClusterNode node) {
        Boolean dataNode = node.attribute("vehicle.service.node");
        return dataNode != null && dataNode;
    }
}

<property name="userAttributes">
    <map key-type="java.lang.String" value-type="java.lang.Boolean">
        <entry key="vehicle.service.node" value="true"/>
    </map>
</property>

<bean class="org.apache.ignite.services.ServiceConfiguration">
    <property name="name" value="VehicleService"/>

    <property name="nodeFilter">
        <bean class="common.filters.VehicleServiceFilter"/>
    </property>
</bean>
Microservices Architecture: Communication

- **In-cluster Communication**
  - Service Grid API
- **Service Interface**
  - Service Nodes
  - Applications
- **Service Implementation**
  - Service Nodes
- **External Protocols**
  - REST, Sockets, etc.

```java
public interface VehicleService extends Service {
    /** Service name */
    public static final String SERVICE_NAME = "VehicleService";

    /**
     * Calls the service to add a new vehicle.
     */
    @param vehicleId Vehicle unique ID.
    @param vehicle Vehicle instance to add.
    public void addVehicle(int vehicleId, Vehicle vehicle);

    /**
     * Calls the service to get details for a specific vehicle.
     */
    @param vehicleId Vehicle unique ID.
    public Vehicle getVehicle(int vehicleId);

    /**
     * Calls the service to remove a specific vehicle.
     */
    @param vehicleId Vehicle unique ID.
    public void removeVehicle(int vehicleId);
}
```
Microservices Architecture: Applications

- **“Internal” Applications**
  - Connect via Ignite API
  - Use Service Grid API
- **“External” Applications**
  - Might not know about the cluster
  - REST, Sockets, etc.
Microservices Architecture: Persistence

• Plugged-in to Data Nodes
  – Ignite Native Persistence
  – RDBMS
  – Cassandra
  – Hadoop
  – Etc.

• Read-through
• Write-through
• Transactional
  – Depends on a storage type
Demo
Resources

• Blog posts:

• GitHub Project With Templates:
  – https://github.com/dmagda/MicroServicesExample

• Service Grid:
  – https://apacheignite.readme.io/docs/service-grid
ANY QUESTIONS?

Thank you for joining us. Follow the conversation.

http://ignite.apache.org

#apacheignite