GeoServer Clustering Revisited

Getting Your Docker On

Derek Kern - Ubisense, Inc
We use GeoServer a lot
Quick Introduction
This talk is a follow up

- I gave a talk entitled “High Performance Geoserver Clusters” at Foss4g NA 2016
  - [https://youtu.be/YvnM2MXrlNg](https://youtu.be/YvnM2MXrlNg)
- The talk concerned the reasoning / processes involved in scaling and clustering GeoServer / GeoWebCache
- We walked through a number of different cluster designs, until most of the design challenges had been accounted for
- This talk shouldn’t have been revelatory. Rather, it followed naturally from the process of addressing the issues that come with clustering GeoServer / GeoWebCache
Final architecture from last year
Final architecture from last year

- This architecture has a single, unified GeoWebCache instance in front the GeoServer instances. It is responsible for caching tiles. Most often, the GeoServer instances are tile generators.
- GeoWebCache uses the load balancer to determine which GeoServer instance will generate the next needed tile.
- This architecture can exploit the maximum amount of tiling capacity from the GeoServer instances.
  - I want to amend this statement → It squeezes the most out of the GeoServer instances, but not necessarily the hardware.
Final architecture from last year

- This architecture had two minor problems
  - GeoWebCache has its own configuration data that must be maintained. Furthermore, this configuration data is dependent upon the configuration of the GeoServer instances
  - A GeoWebCache layer cache must be cleared whenever the associated GeoServer layer undergoes a configuration data change
  - Both of these problems were managed using scripts
Considerations from last year

1. At least one GeoWebCache server is needed so that tiles can be cached and managed. Unified is best
2. A load balancer needs to be in front of GeoServer instances. This load balancer is used by GeoWebCache to determine which GeoServer will fulfill a tile request. It can also be used to service dynamic requests from clients
3. An approach is needed so that GeoServer / GeoWebCache configuration data can be easily managed

* Keep these in mind
What this talk is about

• I wanted to transition the final architecture from last year onto Docker and Docker Swarm. How easy/difficult would it be to transition?
• We will walk through a brief comparison of the two architectures
• We will examine a new Docker-based architecture through the prism of the considerations from last year
• We will also see some benchmark data generated using the Docker-Swarm-based architecture
Docker and Swarm Intro-blurbs
Docker in a small nutshell

• Docker is a very lightweight, containerization technology. Containers do not require hypervisors in that they run directly on the host operating system kernel.
• It provides the ability to package applications, and all required prerequisites, into containers that can be executed in isolation from the host operating system.
• Many containers can be (and typically are) run simultaneously on a single host.
• Overall, the container model offers freedom from the complexity of (1) blending applications within a single OS and (2) fully utilizing available hardware.
Docker Swarm in a small nutshell

- Docker Swarm is used to create and manage clusters of Docker containers. **Kubernetes is an alternative**
- Swarm ‘X’ consists of the compute nodes that have joined ‘X’
  - `docker swarm init` ← Initialize the swarm
  - `docker swarm join` ← A node joins the swarm
- Once a swarm is constituted, services are defined to run within the swarm
  - `docker service create` ← Create a service
    - Minimally, provide a service **name**, **image**, and **replica count** (i.e. number of containers running the provided image)
Docker Swarm routing mesh

- Services are given a port number as well
  - e.g. an Apache service might be defined to port 80
- When a service is given a port number, all nodes in a swarm will respond on that port number by passing requests to the defined service. This occurs whether a node is running a replica for the requested service or not
- The upshot of this is that load balancing is built into Docker Swarm. Load balancing is implied by the fact that the swarm manager is allowed to determine the best node to run a replica
- Consequence: Sticky sessions aren’t available

* An external load balancer can be used
GeoServer, GeoWebCache, and Docker in concert
Docker Swarm Architecture
Architectures side by side
GeoServer Container Definition

Dockerfile

FROM centos:6

RUN yum update -y && yum install -y wget unzip
RUN yum install -y java-1.8.0-openjdk
RUN wget "http://mirror.cogentco.com/.../apache-tomcat-8.5.16.tar.gz" && \
    cd /usr/lib && \
    tar xzvf /tmp/apache-tomcat-8.5.16.tar.gz
RUN wget "https://downloads.sourceforge.net/...geoserver-2.11.1-bin.zip" && \
    cd /usr/lib && \
    unzip /tmp/geoserver-2.11.1-bin.zip

COPY geoserver_data_dir /usr/lib/geoserver-2.11.1/data_dir

ENV JAVA_HOME /usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.131-0.b11.el6_9.x86_64
ENV CATALINA_HOME /usr/lib/apache-tomcat-8.5.16
ENV GEOSERVER_HOME /usr/lib/geoserver-2.11.1

ENTRYPOINT ${GEOSERVER_HOME}/bin/startup.sh

EXPOSE 8080
Through the prism
Are these still problems?

• Reminder: the final architecture from last year had two minor problems
  – GeoWebCache has its own configuration data that must be maintained
  – A GeoWebCache layer cache must be cleared whenever the associated GeoServer layer undergoes a configuration data change

• Answer: Yes, but these problems, again, are soluble using scripting....and some thoughtful Docker image design
Considerations from last year

1. At least one GeoWebCache server is needed so that tiles can be cached and managed. Unified is best

   - GeoWebCache can also be run as a service within the same swarm as GeoServer
   - Note that, typically, GeoServer and GeoWebCache use the same port, 8080. In this architecture, given that each will run within a different service (and the existence routing mesh), they cannot
   - Let GeoServer use port 8081 and GeoWebCache use port 8080
Considerations from last year

2. A load balancer needs to be in front of GeoServer instances. This load balancer is used by GeoWebCache to determine which GeoServer will fulfill a tile request. It can also be used to service dynamic requests from clients.

- By default, load balancing is handled by the Docker Swarm routing mesh.
- An external load balancer can be used.
Considerations from last year

3. An approach is needed so that GeoServer / GeoWebCache configuration data can be easily managed

• There are a variety of approaches
  – #1: A share can be mapped into swarm containers
    • Permissions management can be hairy
  – #2: Start detached GeoServer container to make configuration changes. Export the configuration. Rebuild containers using the configuration data
• I chose option #2
Benchmark
Benchmark

- I spun up three medium instances on Amazon EC2 using the “Amazon Linux AMI 2017.03.1 (HVM), SSD Volume Type” image
- Medium instances have **4 Gb RAM** and **2 Intel Xeon hyperthreaded cores**
- Docker Swarm services
  - db - PostgreSQL 9.6 / PostGIS 2.3 with one replica
  - geoserver - GeoServer 2.11.1 with variable replicas
- Scaling is achieved by increasing the number of replicas within the geoserver service
Benchmark

- The benchmark was performed by:
  - Gathering web requests from GeoServer access logs into a very large file
  - Massaging this data so that it was appropriate for Apache JMeter
  - Building JMeter configurations for each of the GeoServer cluster configurations
  - Running JMeter for each configuration over the request file
Benchmark - Swarming

Display of the swarm nodes

<table>
<thead>
<tr>
<th>ID</th>
<th>HOSTNAME</th>
<th>STATUS</th>
<th>AVAILABILITY</th>
<th>MANAGER STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>93smex2pyf2x6r435syjw2b4z</td>
<td>ip-10-0-0-227</td>
<td>Ready</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>lurqbjr2thhat923btf7d2q1r  *</td>
<td>ip-10-0-0-249</td>
<td>Ready</td>
<td>Active</td>
<td>Leader</td>
</tr>
<tr>
<td>md0qrrfayj03kbm8z6nrlrdjx</td>
<td>ip-10-0-0-233</td>
<td>Ready</td>
<td>Active</td>
<td></td>
</tr>
</tbody>
</table>

Display of the db service with one replica - It is running on node 249

```
[ec2-user@ip-10-0-0-249 postgresql_postgis_server]$ docker service ps db
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>IMAGE</th>
<th>NODE</th>
<th>DESIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>bn10l18y2aer</td>
<td>db.1</td>
<td>34.200.171.72:5000/postgresql_postgis_server:1</td>
<td>ip-10-0-0-249</td>
<td>Running</td>
</tr>
</tbody>
</table>

Display of the geoserver service with one replica - It is running on node 233

```
[ec2-user@ip-10-0-0-249 postgresql_postgis_server]$ docker service ps geoserver
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>IMAGE</th>
<th>NODE</th>
<th>DESIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>vu04ylunkkx4</td>
<td>geoserver.1</td>
<td>34.200.171.72:5000/geoserver_server:1</td>
<td>ip-10-0-0-233</td>
<td>Running</td>
</tr>
<tr>
<td>rbrvwcprijh3</td>
<td>geoserver.2</td>
<td>34.200.171.72:5000/geoserver_server:1</td>
<td>ip-10-0-0-227</td>
<td>Running</td>
</tr>
<tr>
<td>8mgjzcnp753l</td>
<td>geoserver.3</td>
<td>34.200.171.72:5000/geoserver_server:1</td>
<td>ip-10-0-0-249</td>
<td>Running</td>
</tr>
</tbody>
</table>

Display of the geoserver service with three replicas - One is running on node 249

```
[ec2-user@ip-10-0-0-249 postgresql_postgis_server]$ docker service ps geoserver
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>IMAGE</th>
<th>NODE</th>
<th>DESIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>5gl77ne4zvil</td>
<td>geoserver.1</td>
<td>34.200.171.72:5000/geoserver_server:1</td>
<td>ip-10-0-0-233</td>
<td>Running</td>
</tr>
<tr>
<td>rbrvwcprijh3</td>
<td>geoserver.2</td>
<td>34.200.171.72:5000/geoserver_server:1</td>
<td>ip-10-0-0-227</td>
<td>Running</td>
</tr>
<tr>
<td>8mgjzcnp753l</td>
<td>geoserver.3</td>
<td>34.200.171.72:5000/geoserver_server:1</td>
<td>ip-10-0-0-249</td>
<td>Running</td>
</tr>
</tbody>
</table>
Benchmark - Results

- The performance jump from 1 to 2 replicas is substantial. It is almost 2X.
- The performance jump from 2 to 3 replicas is less substantial. This is likely due to a replica sharing the node running the db service replica.
- The performance slumps from 5 to 6 replicas. At this point, all nodes have 2 replicas and one is also running the db replica.
Wrap up
Conclusion

• It was shockingly easy to create a usable GeoServer / GeoWebcache cluster, e.g. the benchmark construction took less than 3 hours
• Docker and Swarm allow us to take the final architecture from last year and stretch it to more fully utilize available hardware
• For this application, Docker and Swarm introduce no significant difficulties and greatly simplify “scaling-in” to available hardware
• Swarm provides the ability to dynamically scale a GeoServer cluster to match usage
Thank you!

FIND OUT MORE
Derek Kern
Principal Architect
Email: derek.kern@ubisense.net
www.ubisense.net