JUDCon

JBoss Users & Developers Conference

2010:Berlin

Running a JBoss cluster in the cloud

Bela Ban

“JBoss Clustering uses IP multicasting, so it doesn't work on EC2 !@#$@”

WRONG !

Of course it DOES !

We'll look at the 5 different ways of running a JBoss cluster in the cloud

Agenda

- Clouds and IP multicasting
- The discovery problem
- JGroups as cluster communication backbone
- The different discovery configs
  - Static, lookup service, shared directory, S3, database
- Demo

Re: demo

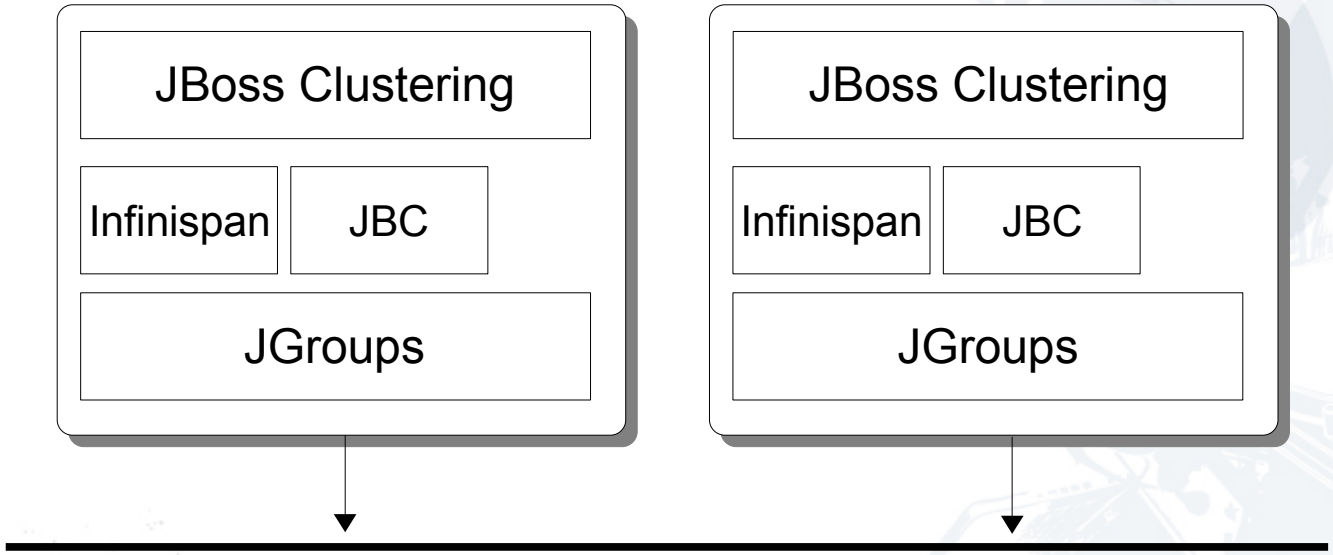
- If you want to participate in the demo later, download it from http://www.jgroups.org → Downloads → JUDCon2010Demo
- To run it:
  - java -jar JUDConDemo.jar -host x.x.x.x -user *yourname*

So why doesn't a JBoss cluster run out of the box in the cloud ?

- JBoss clustering uses *IP multicasting* by default
  - It is the simplest way to discover nodes in a cluster, no configuration required
  - Most folks run clusters off of a single switch
  - Cluster nodes immediately find each other
- However: most cloud hosters don't support IP multicasting !
- So let's take a look at the alternatives

Before we do that, let's take a step back and look at the architecture of JBoss clustering

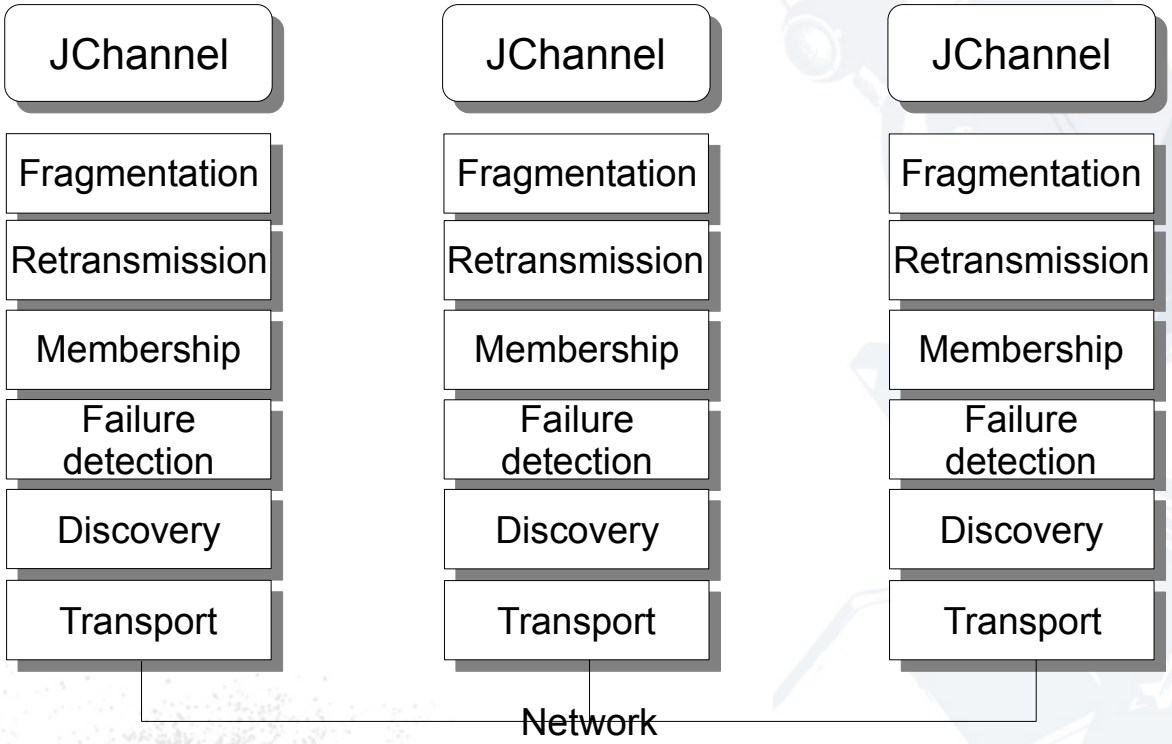
A cluster in JBoss



JGroups

- Reliable cluster transport
- Tasks
  - Discovers nodes in a cluster
  - Joins new nodes, removes left or crashed nodes
  - Retransmission, ordering, duplicate removal
  - Transports: UDP (IP multicasting), TCP

JGroups architecture



So where in JBoss are the configs located ?

… and how do I tell JBoss to use a specific configuration ?

Cluster configuration

- All configs are in one XML file:
  - JBOSS/server/CONFIG/deploy/cluster/jgroups-channelfactory.sar/META-INF/jgroups-channelfactory-stacks.xml
- We have configs for UDP and TCP

Sample configuration

```
<protocol_stacks>
    <stack name="udp">
        <config>
          <UDP bind_port="${jboss.jgroups.udp.bind_port:55200}" />
          <PING timeout="2000" num_initial_members="3"/>
          <MERGE2 max_interval="100000" min_interval="20000"/>
          <FD_SOCK/>
          <FD timeout="6000" max_tries="5"/>
          <VERIFY_SUSPECT timeout="1500"/>
          <BARRIER/>
          <pbcast.NAKACK retransmit_timeout="300,600,1200" />
          <UNICAST timeout="300,600,1200,2400,3600"/>
          <pbcast.STABLE desired_avg_gossip="50000" max_bytes="400k"/>
          <pbcast.GMS join_timeout="3000" />
          <FC max_credits="2m" min_threshold="0.10" />
          <FRAG2 frag_size="60k"/>
        </config>
    </stack>

    <stack name="tcp">
        <config>
            <TCP start_port="${jboss.jgroups.tcp.tcp_port:7600}"/>
            <TCPPING timeout="3000"
                     initial_hosts="Host-A[7600],Host-B[7600]”/>
             ...
        </config>
    </stack>
</protocol_stacks>
```

How to start JBoss with a specific config

- Pass a system property to run.sh:
  - run.sh -Djboss.default.jgroups.stack=tcp
    - Voila: we run a TCP based stack now !
- This is how we're going to start a JBoss cluster in the cloud
- Of course, we could also create a virtual image (e.g. an AMI) with a hard coded config

What are the 5 different discovery configurations ?

Method #1: static list of nodes

- Provide a list of the cluster nodes:

```
<TCP … />
<TCPPING initial_hosts="192.168.1.5[7800],192.168.1.3[7800]" />
```

- However, we don't know the IP address of a node before startup...
  - Use elastic IP addresses (EC2)
  - Map IP address to an ad-hoc DNS (dyndns.org)

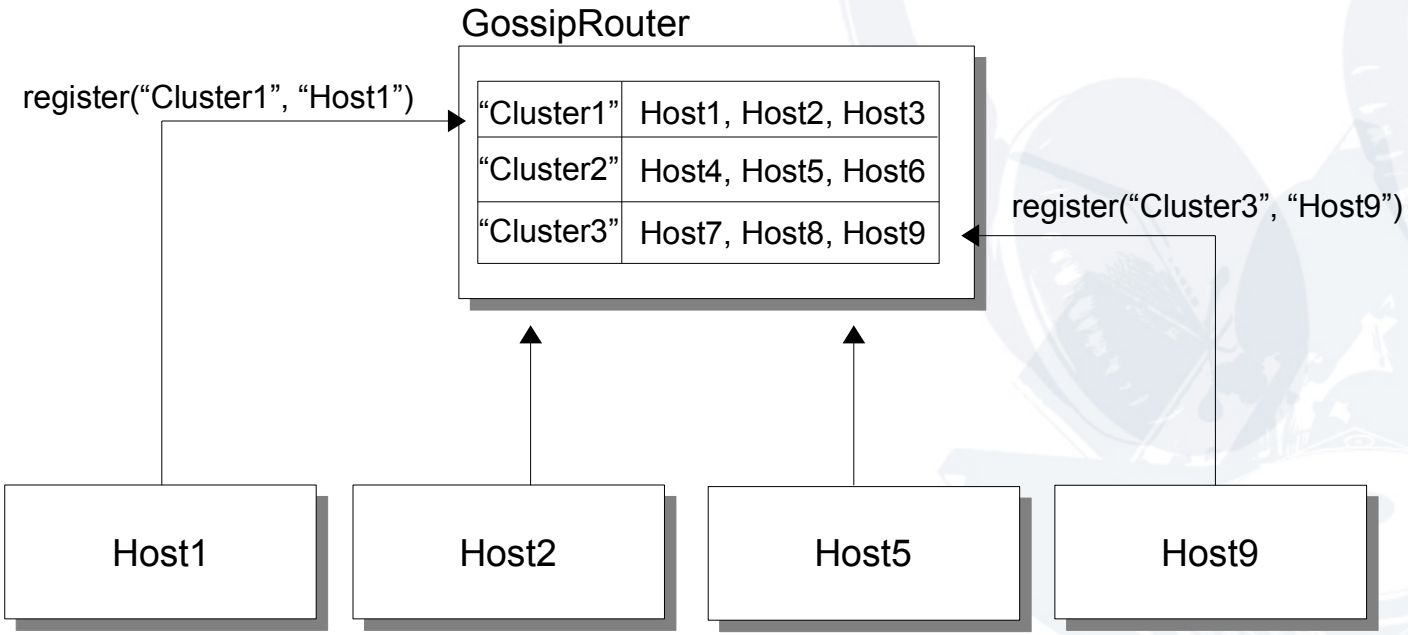
Method #2: use a lookup service

- Each node registers with a lookup service

```
<TCP … />
<TCPGOSSIP initial_hosts="http1.dyndns.org[12001]" />
```

- We ask the lookup service for a list of cluster nodes
- There can be multiple lookup services
- Disadvantage: an external process

Lookup service architecture



Method #3: place node info into a shared directory

- Discovery done through parsing of files in a directory
- 1 directory per cluster, 1 file per node
  - Directory name == cluster name
- For cluster discovery, the directory should be on a shared drive (e.g. NFS)
- Config:

```
<TCP … />
<FILE_PING location="/mnt/nas/jgroups" />
```

Method #4: place node info in an S3 bucket

- EC2 specific, 'location' == bucket name
  - Bucket name needs to be unique !
- With access_key and secret_access_key
  - Can be null if bucket is public

```
<TCP … />
<FILE_PING location="jgroups" acccess_key="xxx"
           secret_access_key="xxx" />
```

- We can also generate unique buckets

```
<TCP … />
<FILE_PING location="jgroups" prefix="jgroups-2.11"
           acccess_key="xxx" secret_access_key="xxx" />
```

Method #5: place node info into a database

- Assumes we have a DB somewhere, accessible by all cluster nodes
  - Node info stores in a table
  - Table name == cluster name
- Not yet done:
  - https://jira.jboss.org/browse/JGRP-1231

Conclusion

- There are 5 different ways of running a JBoss cluster in the cloud !
- JBoss-supplied virtual instances (StormGrind, CirrAS) use the presented discovery mechanisms

Links

- JGroups: jgroups.org
- StormGrind: jboss.org/stormgrind
- JBoss appliances: http://community.jboss.org/wiki/CirrASAppliances