

Vert.x Clustering

Felix Hamann

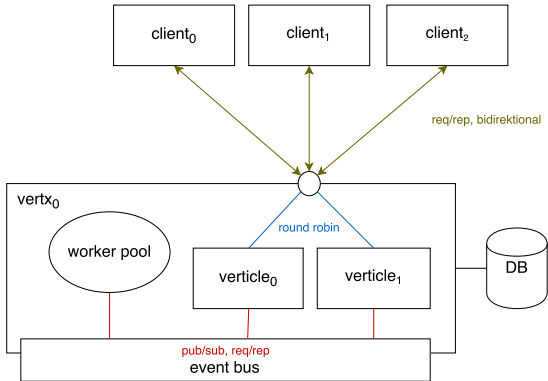
Was ist Clustering?

- ❖ Ein Zusammenschluss homogener Computer
- ❖ Jeder Computer erfüllt die gleiche Aufgabe
- ❖ In der Regel per lokalem Netzwerk verbunden
- ❖ Verhält sich wie ein großer Rechner

Warum clustern?

- ❖ Ausfallsicherheit erhöhen
- ❖ Für Hochverfügbarkeitssysteme
- ❖ Um Supercomputer zu bauen
- ❖ Variabel auf Lastschwankungen reagieren (IaaS/PaaS)
- ❖ Kann kosteneffizient sein

Vert.x Architektur



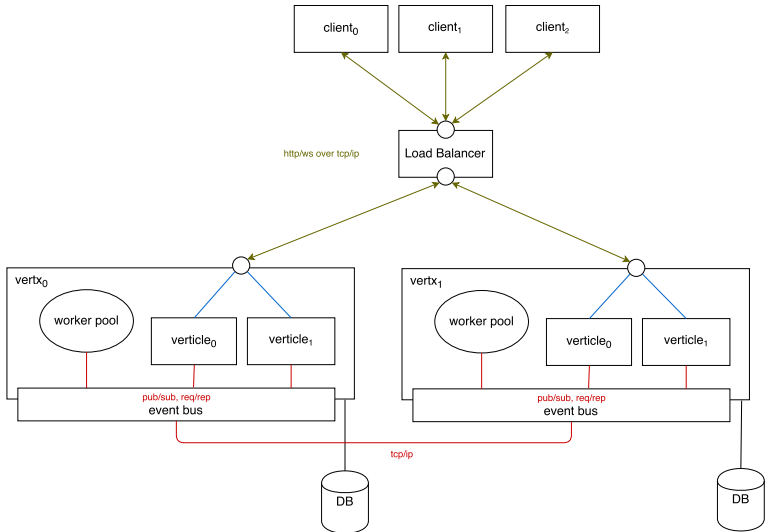
Clustering bei Vert.x

1. Optionen setzen
2. Vert.x asynchron starten
3. Profit

```
VertxOptions opts = new VertxOptions();  
opts.setClusterPort(CLUSTER_PORT);  
opts.setClusterHost(CLUSTER_HOST);
```

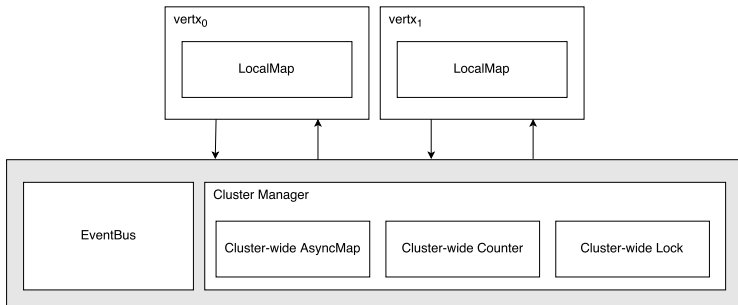
```
Vertx.clusteringVertx(opts, ar -> {  
    final Vertx vertx = ar.result()  
    // deploy verticles  
});
```

Vert.x Cluster Beispiel-Architektur



Shared Data

1. LocalMap pro Instanz sichtbar
2. AsyncMap pro Cluster
3. Atomare Counter
4. Verteilte Locks



Local Shared Map

Synchrones Interface zum Austausch von unveränderlichen Daten zwischen verschiedenen Verticle-Instanzen

```
final SharedData sd = this.vertx.SharedData();  
LocalMap<String, String> local = sd.getLocalMap("mapname");  
  
local.put("foo", "bar");  
local.get("foo"); // "bar"
```


Cluster Wide Map

Asynchrones Interface um Daten quer über den gesamten Cluster auszutauschen. Kann zB. verwendet werden um Nutzersessions zu verwalten.

```
final SharedData sd = this.vertx.SharedData();
sd.<String , String>getClusterWideMap("name", ar -> {

    final AsyncMap<String , String> map = ar.result();
    map.put("foo", "bar", putResult -> {
        map.get("foo", getResult -> {
            String val = getResult.result(); // "bar" ?
        }
    });
});
```