(Distributed Cache)

How to create a simple Spring Boot application, using Hazelcast as a Distributed Cache. This can help improve database performance.

by Diraj Pandey \cdot Feb. 25, 21 \cdot Java Zone \cdot Tutorial Comment (0) Like (2) ☆ Save **Tweet**

Join the DZone community and get the full member experience. **JOIN FOR FREE High Performance Websockets on Android** DZone. Building high performance, real-time applications on mobile devices comes with a unique set of **ON-DEMAND** challenges. In this webinar, we'll explore a high performance Websocket application built with Netty, WEBINAR and we'll use Mason to deploy it at scale on a bank of Android tablets. Watch now ▶ MASON

Hazelcast provides central and predictable scaling of applications via in-memory access to frequently used

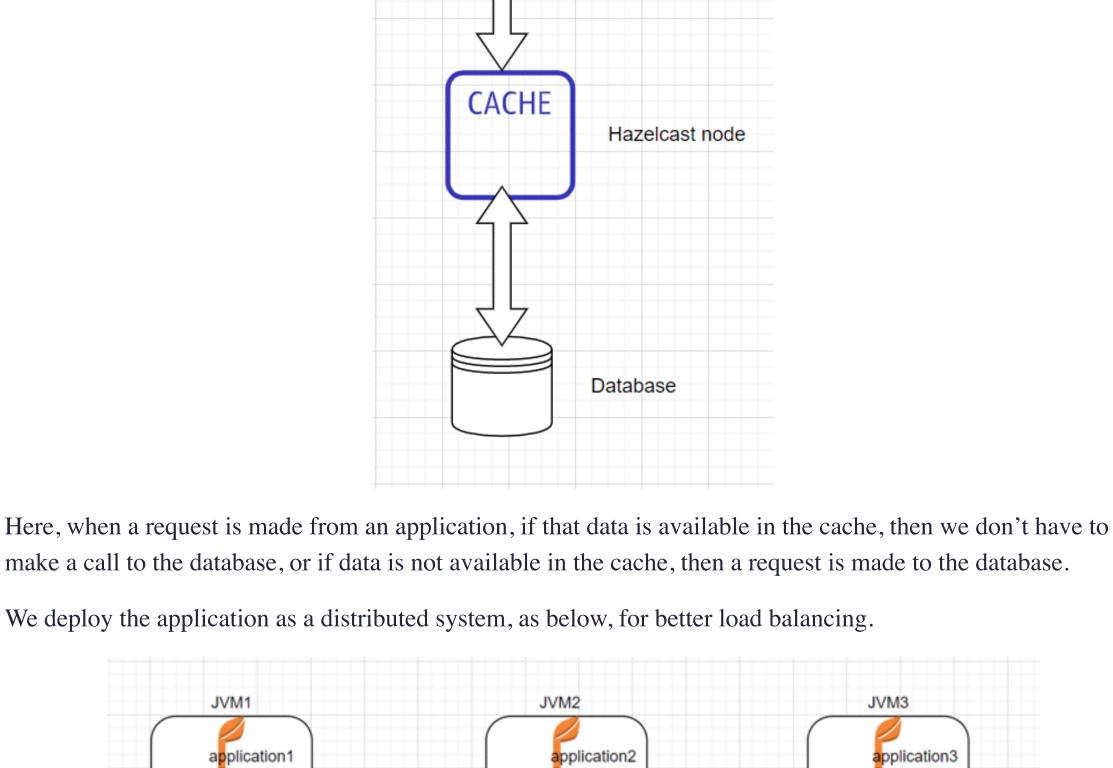
data and across an elastically scalable data grid. These techniques both reduce the query load on databases and

Spring Boot is often used to create stand-alone, production-grade Spring-based Applications that you can 'just

run.' Let us create a simple Spring Boot application with Hazelcast as a Distributed Cache. Below is an implementation of a local cache for better performance.

Application

help improve application speed.



JVM3

CACHE

application3

Cache node Cache node CRUD opertions CRUD opertions

CACHE

CRUD opertions

JVM1

CACHE

1 <plugin>

</plugin>

pom.xml

XML

9

10

11

12

13

53

54

Java

16

18

21 22

23

24 25

30 31

}

</plugins>

</build>

1 package com.rbitcs;

3 import java.util.HashMap;

10 import com.hazelcast.config.Config;

12 import com.hazelcast.core.Hazelcast;

14 import com.hazelcast.core.IMap;

17 import com.rbitcs.model.Doctors;

20 public class HospitalApplication {

19 @SpringBootApplication

6 import org.springframework.boot.SpringApplication;

8 import org.springframework.context.annotation.Bean;

11 import com.hazelcast.config.ManagementCenterConfig;

public static void main(String[] args) {

public void setName(String name) {

public void setAddress(String address) {

public void setBalance(long balance) {

this.name = name;

return address;

public String getAddress() {

this.address = address;

this.balance = balance;

public long getBalance() {

return balance;

36 37

38 39

41

42 43 44

45

46 47

48

53

18

22 23

24

25 26

27 28

29

30

31

32

33

database.

19 @RestController

@Autowired

@Autowired

20 @RequestMapping(path = "doctors") 21 public class DoctorController {

private DoctorRepository doctorRepository;

@GetMapping(path = { "/get/{doctorNumber}" })

public Doctor getDoctor(@PathVariable("doctorNumber") String doctorNumber) {

: DoctorRepository.findByDoctorNumber(doctorNumber);

/First call is to check if doctormap has doctor details if yes, return the value otherwise call

Doctor doctor = (doctortMap.get(doctorNumber) != null) ? doctorMap.get(doctorNumber)

private Map<String, Doctor> doctorMap;

13 import com.hazelcast.core.HazelcastInstance;

15 import com.codeusingjava.model.UserAccount;

9 import org.springframework.context.annotation.Configuration;

7 import org.springframework.boot.autoconfigure.SpringBootApplication;

SpringApplication.run(HospitalApplication.class, args);

4 import java.util.Map;

55 </project>

<configuration>

</configuration>

POM file should look like below:

<parent>

</parent>

<version>0.0.1-SNAPSHOT

<version>2.2.5.RELEASE

<groupId>org.springframework.boot

<artifactId>spring-boot-starter-parent</artifactId>

<relativePath /> <!-- lookup parent from repository -->

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<mainClass>com.rbitcs.DoctorApplication</mainClass>

application1

```
DataBase
Implementation
Here is the implementation of a simple spring boot application.
First, we make changes in the pom file by adding a Hazelcast dependency, as below:
 XML
  1 <dependency>
        <groupId>com.hazelcast
        <artifactId>hazelcast</artifactId>
  4 </dependency>
We also add a plugin where the entry point to the application is provided in the distributed system. Since
DoctorApplication is the main class it is added in the plugin as below:
 XML
```

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</modelVersion> <groupId>com.rbitcs</groupId> <artifactId>hazelcast-springboot</artifactId>

```
15
      cproperties>
16
          project.build.sourceEncoding>UTF-8
          oject.reporting.outputEncoding>UTF-8
17
18
          <java.version>1.8</java.version>
19
      </properties>
20
21
      <dependencies>
22
          <dependency>
23
             <groupId>org.springframework.boot</groupId>
24
              <artifactId>spring-boot-starter-web</artifactId>
25
          </dependency>
26
27
          <dependency>
28
              <groupId>org.springframework.boot
29
             <artifactId>spring-boot-starter-data-jpa</artifactId>
30
          </dependency>
31
32
          <dependency>
33
              <groupId>mysql</groupId>
34
              <artifactId>mysql-connector-java</artifactId>
35
              <scope>runtime</scope>
36
          </dependency>
37
38
          <dependency>
39
              <groupId>com.hazelcast
40
              <artifactId>hazelcast</artifactId>
41
          </dependency>
42
      </dependencies>
      <build>
43
45
          <plugins>
46
              <plugin>
47
                 <groupId>org.springframework.boot
48
                 <artifactId>spring-boot-maven-plugin</artifactId>
49
                 <configuration>
50
                     <mainClass>com.RBITCS.DoctorApplication</mainClass>
51
                 </configuration>
             </plugin>
52
```

We will be using IMAP provided by hazelmap as the cache. IMAP implements a map interface. Below is the

Spring Boot application where beans use IMAP for caching in configuring the Hazelcast instance.

26 @Bean 27 public Config hazelCastConfig() { 28 return new Config().setManagementCenterConfig(new ManagementCenterConfig().setEnabled(true).setUrl("http://localhost:8080/hazelcast-29 mancenter"));

```
32
 33
        @Bean
        public HazelcastInstance hazelcastInstance(Config hazelCastConfig) {
 34
            return Hazelcast.newHazelcastInstance(hazelCastConfig);
 35
 36
 37
 38
        @Bean
 39
        public Map<String, Doctor> doctorMap(HazelcastInstance hazelcastInstance) {
 40
            return hazelcastInstance.getMap("doctorMap");
 41
Since we are making use of IMAP we need to serialize the object. Here, Doctor object should be serializable.
 Java
  1 package com.rbitcs.model;
  3 import java.io.Serializable;
  5 import javax.persistence.Column;
  6 import javax.persistence.Entity;
  7 import javax.persistence.GeneratedValue;
  8 import javax.persistence.GenerationType;
  9 import javax.persistence.Id;
 10 import javax.persistence.Table;
 11
 12 @Entity
 13 @Table(name = "doctor")
 14 public class Doctor implements Serializable {
 15
 16
        private static final long serialVersionUID = 1L;
 17
 18
        @Id
 19
        @Column(name = "id")
        @GeneratedValue(strategy = GenerationType.IDENTITY)
 20
        private Long id;
 21
 22
 23
        @Column(name = "doctorNumber")
 24
        private String doctorNumber;
 25
        @Column(name = "name")
 26
 27
        private String name;
 28
 29
        @Column(name = "address")
 30
        private String address;
 31
 32
        public String getName() {
 33
            return name;
 34
 35
```

```
54
 55
 56
        public String getDoctorNumber() {
 57
            return doctorNumber;
 58
 59
 60
        public void setDoctorNumber(String doctorNumber) {
 61
            this.doctorNumber = doctorNumber;
 62
 63 }
Let us create a controller and auto-wire doctorMap:
 Java
  1 package com.rbitcs.controller;
  3 import java.util.List;
  4 import java.util.Map;
  5 import java.util.Optional;
  7 import org.springframework.beans.factory.annotation.Autowired;
  8 import org.springframework.web.bind.annotation.DeleteMapping;
  9 import org.springframework.web.bind.annotation.GetMapping;
 10 import org.springframework.web.bind.annotation.PathVariable;
 11 import org.springframework.web.bind.annotation.PostMapping;
 12 import org.springframework.web.bind.annotation.RequestBody;
 13 import org.springframework.web.bind.annotation.RequestMapping;
 14 import org.springframework.web.bind.annotation.RestController;
 15
 16 import com.rbitcs.db.DoctorRepository;
 17 import com.rbitcs.model.Doctor;
```

```
34
             return doctor;
        }
 35
 36
 37
        @PostMapping("/add")
 38
        public void createDoctor(@RequestBody Doctor doctor) {
            //save doctor details in cache
 39
            doctorMap.put(doctor.getDoctorNumber(), doctor);
 40
            doctorRepository.save(doctor);
 41
 42
        }
 43
        @DeleteMapping(path = { "/delete/{doctorNumber}" })
 44
        public Doctor deleteDoctor(@PathVariable("doctorNumber") String doctorNumber) {
 45
            //remove doctor details from both cache and database
 46
            doctorMap.remove(doctorNumber);
 47
            return doctorRepository.deleteByDoctorNumber(doctorNumber);
 48
 49
 50 }
Start Application
Nice! We have made all the necessary changes.
Let us start the application with the below commands:
 java -Dserver.port='8081' -jar boot-hazel-0.0.1-SNAPSHOT.jar
 java -Dserver.port='8082' -jar boot-hazel-0.0.1-SNAPSHOT.jar
 java -Dserver.port='8083' -jar boot-hazel-0.0.1-SNAPSHOT.jar
Let's say we have started 3 instances at 3 different ports (8081, 8082, 8083).
When a doctor's details are stored via the 8081 port application, then the same object can be retrieved from the
8083 port or 8082 port application from the cache and it does not need to make a call to the database.
After the above implementation, the architecture looks like below:
                                                 JVM2
                                                                              JVM3
                     JVM1
```

Here all the nodes are synchronized and the same date will be retrieved when requested from any of the JVM with better performance and better load balance. 2021 CI/CD Trend Report

CACHE

CRUD opertions

DataBase

Automation for Reliable Software Delivery: In DZone's newest Trend Report, we dive deeper into the latest DevOps practices that are advancing continuous integration, continuous delivery, and release

CRUD opertions

Opinions expressed by DZone contributors are their own. Popular on DZone • MySQL to Redshift: 4 Ways to Replicate Your Data

• How to Configure log4j2 in Spring Boot Application Using log4j2 .Properties? | Spring Boot Logging

with Wrike.

Presented by Wrike

wrike

ABOUT US

MVB Program

Terms of Service

Privacy Policy

LEGAL

CONTRIBUTE ON DZONE

Become a Contributor

Visit the Writers' Zone

Article Submission Guidelines

Careers Sitemap

REND REPORT

application1

CRUD opertions

automation. Download the report ▶

Topics: HAZELCAST, TUTORIAL, SPRING BOOT, SPRING BOOT APPLICATION

Apache Kafka for Industrial IoT and Manufacturing 4.0

• Let's Use Optional to Fix Our Method Contract

DZone.

Java Partner Resources Monday.com helps teams work more efficiently to DZone. execute projects that deliver results on time. Try for

Free > TREND REPORT Presented by Monday.com

Deliver more of your best work, faster,

Wrike is an award-winning work management

software that enables teams to plan and track

reports. Start for free today! ▶

projects, collaborate in real-time, and automate Presented by MariaDB

DZone.

REFCARD

2021 CI/CD Trend Report

Presented by **DZone**

Dive into the latest DevOps practices that are

advancing continuous integration, continuous

delivery, and release automation. Download ▶

Getting Started With Distributed SQL

Learn the key characteristics of distributed SQL databases, how functionality compares across

vendors' offerings, and more! Read now ▶

application3

Hażelcast

CACHE

Let's be friends: 🔊 🕥 f in

DZone.com is powered by

asana

All Your Work In One Place

are located. Try for free ▶

Presented by Asana

AnswerHub

Asana helps teams orchestrate their work, from

accomplish more with less, no matter where they

daily tasks to strategic initiatives. With Asana,

teams are more confident, move faster, and

ADVERTISE About DZone Developer Marketing Blog Send feedback **Advertise with DZone** +1 (919) 238-7100

CONTACT US

Suite 150

600 Park Offices Drive

support@dzone.com

+1 (919) 678-0300

Research Triangle Park, NC 27709

1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

Couchbase

Q