Hazel Cast

ارائه دهنده: مهدی رسول زاده

درس: برنامه سازی وب - نیمسال اول 1403

Hazel cast چیست؟

Hazelcast یک پلتفرم قوی و Open Source برای شبکه‌سازی داده‌ها در حافظه (In-Memory Data Grid) است که ساختارهای داده توزیع شده و ابزارهای محاسباتی را برای مدیریت و پردازش داده‌ها در یک خوشه کامپیوتر فراهم می‌کند.
Hazelcast می‌تواند چندین نمونه از اعضای خوشه را روی همان JVM اجرا کند که به طور خودکار اعضای جدید را به خوشه اضافه می‌کند.

به طور کلی، Hazelcast به عنوان یک شبکه داده‌های درون حافظه‌ای عمل می‌کند و از نرم‌افزار توزیع شده در یک خوشه کامپیوتر بهره می‌برد که به طور جمعی حافظه خود را برای دسترسی به داده‌های مشترک به اشتراک می‌گذارند. این پلتفرم با تکثیر داده‌های ذخیره شده، دسترسی به داده‌ها را افزایش داده و پردازش را تسریع می‌کند.

ویژگی های :Hazelcast

- محاسبات توزیع شده
- تقسیم بندی داده‌ها با پشتیبان گیری
- تکرار در شبکه گسترده ((WAN
- جستجو و ایندکس گذاری
- پردازش جریان

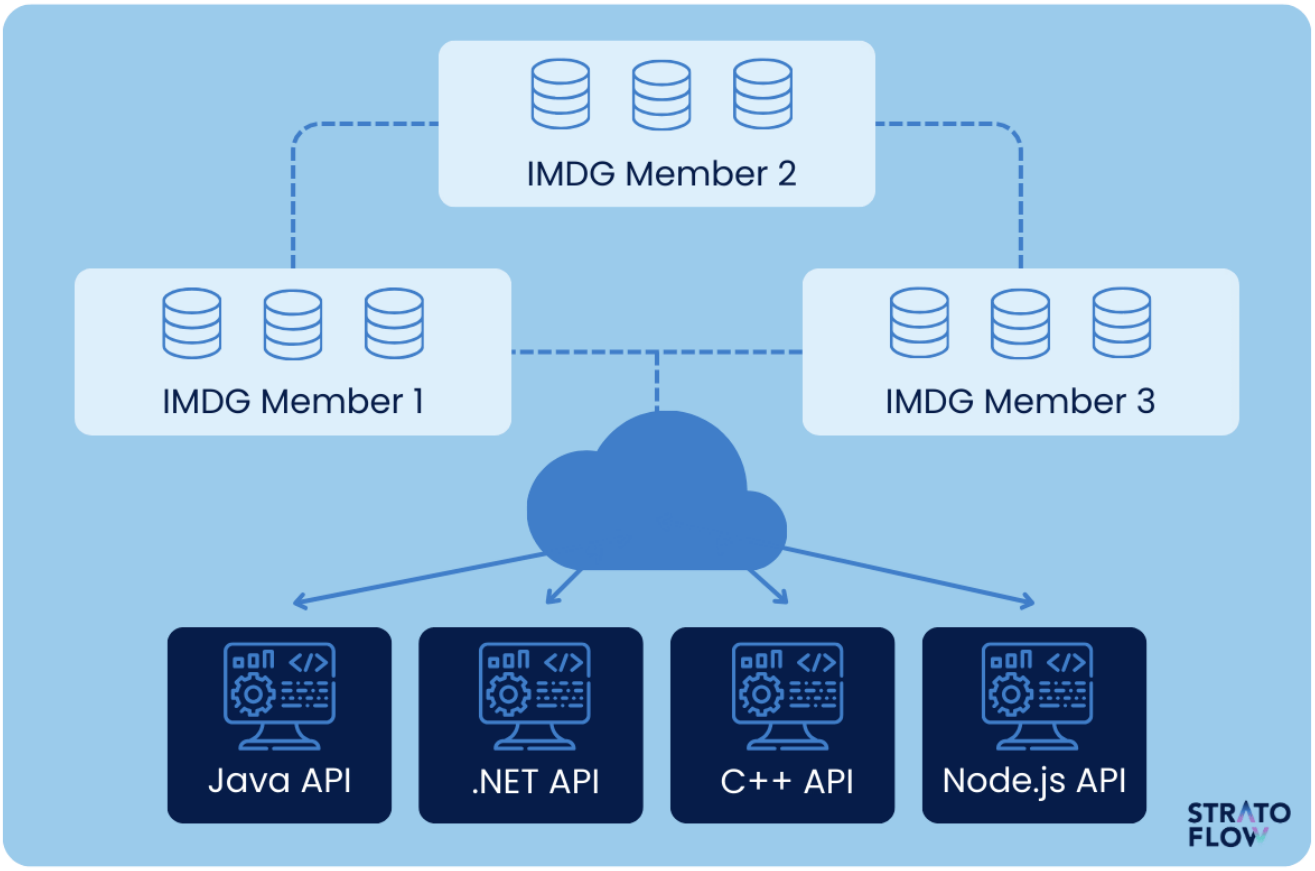
معماری Hazelcast

معماری Hazelcast به گونه‌ای است که وقتی یک عضو جدید به خوشه اضافه می‌شود یا یکی از اعضا خوشه را ترک می‌کند، داده‌ها را دوباره تقسیم بندی می‌کند و داده‌ها را با استفاده از الگوریتم hash به پارتیشن‌ها اختصاص می‌دهد.

اعضای خوشه در معماری Hazelcast به عنوان واحدهای محاسباتی و ذخیره‌سازی عمل می‌کنند و به قابلیت‌های ارتباطی و اشتراک‌گذاری داده‌ها در خوشه Hazelcast کمک می‌کنند تا انعطاف‌پذیری و عملکرد را افزایش دهند.

مرکز مدیریت Hazelcast می‌تواند برای نظارت بر وضعیت کلی خوشه‌ها، تحلیل و مرور ساختارهای داده، به‌روزرسانی پیکربندی نقشه‌ها، و گرفتن نخریزی از نودها استفاده شود.

Hazelcast از استراتژی تکثیر استفاده می‌کند که کپی‌های اصلی و پشتیبان پارتیشن‌ها را به طور یکنواخت بین اعضای خوشه توزیع می‌کند تا از تکرارپذیری و مقیاس‌پذیری اطمینان حاصل شود. Hazelcast از طریق زیربخش (CP Consistency) که به طور خاص برای ساختارهایی که نیاز به دقت بالا دارند طراحی شده است، از یکپارچگی و ثبات داده‌ها اطمینان حاصل می‌کند. همچنین برای سازه‌هایی که نیاز به دقت کمتری دارند، به دنبال دستیابی به بهترین تلاش ممکن است.



یک مثال ساده

```
Map<Integer, String> mapCustomers = Hazelcast.getMap("customers");
mapCustomers.put(1, "Joe");
mapCustomers.put(2, "Ali");
mapCustomers.put(3, "Avi");

System.out.println("Customer with key 1: "+ mapCustomers.get(1));
```

```
System.out.println("Map Size:" + mapCustomers.size());

Queue<String> queueCustomers = Hazelcast.getQueue("customers");
queueCustomers.offer("Tom");
queueCustomers.offer("Mary");
queueCustomers.offer("Jane");
System.out.println("First customer: " + queueCustomers.poll());
System.out.println("Second customer: "+ queueCustomers.peek());
System.out.println("Queue size: " + queueCustomers.size());
```

این یک مثال ساده استفاده از hazelcast است. در ادامه استفاده آن در client API نیز گفته شده است.

```
ClientConfig clientConfig = new ClientConfig();
clientConfig.addAddress("127.0.0.1:5701");
HazelcastInstance client = HazelcastClient.newHazelcastClient(clientConfig);
IMap map = client.getMap("customers");
System.out.println("Map Size:" + map.size());
```

بکارگیری hazelcast در spring boot

ابتدا وابستگی‌ها (dependencies) ها را به پروژه اضافه میکنیم.

Maven:

```
<dependency>
     <groupId>com.hazelcast</groupId>
     <artifactId>hazelcast-spring</artifactId>
     <version>4.2</version> <!-- Use the latest version applicable --
></dependency>
```

سپس config را به اپلیکیشن اضافه میکنیم.

```
private static Config createConfig() {
  Config config = new Config();

  // Create a MapConfig for your map
```

```
        MapConfig mapConfig = new MapConfig("myMap")
            .setTimeToLiveSeconds(360) // Set TTL to 360 seconds
            .setMaxIdleSeconds(200); // Set Max Idle to 200 seconds

        // Add the MapConfig to the configuration
        config.addMapConfig(mapConfig);

        return config;
    }
```

اضافه کردن یک سریالایزر (serializer) و کنترلر سفارشی برای تعامل با داده‌های کش در Hazelcast، برای بهینه سازی عملکرد و مدیریت موثر داده‌ها حیاتی است.

Hazelcast به طور پیش‌فرض از سریال‌سازی جاوا استفاده می‌کند، اما این روش سریال سازی چندان کارآمد نیست زیرا به کندی عمل می‌کند و فرم‌های سریال‌سازی بزرگی تولید می‌کند. یک سریالایزر سفارشی می‌تواند عملکرد Hazelcast را به طور قابل توجهی بهبود بخشد و زمان و اندازه سریال‌سازی را کاهش دهد، که برای سیستم‌های توزیع‌شده که داده‌ها به‌طور مکرر از طریق شبکه منتقل می‌شوند، بسیار مهم است.

کنترلر، از سوی دیگر، معمولاً بخشی از کد برنامه شماست که تعاملات کش را مدیریت می‌کند. این کنترلر تضمین می‌کند که کش به طور موثر استفاده می‌شود و عملیات‌هایی مانند خواندن و نوشتن در کش، بی‌اعتبار کردن کش، و هماهنگ‌سازی را انجام می‌دهد.

یک نمونه از کنترلر:

```
@RestController

public class CommandController {

       @Autowired private HazelcastInstance hazelcastInstance;

       private ConcurrentMap<String,String> retrieveMap() {

        return hazelcastInstance.getMap("map");

       }

        @PostMapping("/put")

       public CommandResponse put(@RequestParam(value = "key") String key,
       @RequestParam(value = "value") String value) {

       retrieveMap().put(key, value);

       return new CommandResponse(value);
```

```
	}

	@GetMapping("/get")

	public CommandResponse get(@RequestParam(value = "key") String key) {

	String value = retrieveMap().get(key);

	return new CommandResponse(value);

	}

}
```

حالا با دستور زیر برنامه را می‌توانید اجرا کنید.

```
mvn spring-boot:run -Dspring-boot.run.jvmArguments="-Dserver.port=8080"
```

حال در یک پورت دیگر برنامه را دوباره اجرا کنید. (مثلا 8081)

در خروجی باید hazelcast cluster را به فرم زیر مشاهده کنید:

```
Members {size:2, ver:2} [

	Member [192.168.1.64]:5701 - 520aec3f-58a6-4fcb-a3c7-498dcf37d8ff
	Member [192.168.1.64]:5702 - 5c03e467-d457-4847-b49a-745a335db557 this

]
```

منابع:

hazelcast doc

startoflow