Index

Overview

Apache Geode is a data management platform that provides real-time, consistent access to data-intensive applications throughout widely distributed cloud architectures.

Geode pools memory, CPU, network resources, and optionally local disk across multiple processes to manage application objects and behavior. It uses dynamic replication and data partitioning techniques to implement high availability, improved performance, scalability, and fault tolerance. In addition to being a distributed data container, Apache Geode is an in-memory data management system that provides reliable asynchronous event notifications and guaranteed message delivery.

Apache Geode is a mature, robust technology originally developed by GemStone Systems. Commercially available as GemFire™, it was first deployed in the financial sector as the transactional, low-latency data engine used in Wall Street trading platforms. Today Apache Geode technology is used by hundreds of enterprise customers for high-scale business applications that must meet low latency and 24x7 availability requirements.

Main Concepts and Components

Caches are an abstraction that describe a node in a Geode distributed system.

Within each cache, you define data *regions*. Data regions are analogous to tables in a relational database and manage data in a distributed fashion as name/value pairs. A *replicated* region stores identical copies of the data on each cache member of a distributed system. A *partitioned* region spreads the data among cache members. After the system is configured, client applications can access the distributed data in regions without knowledge of the underlying system architecture. You can define listeners to receive notifications when data has changed, and you can define expiration criteria to delete obsolete data in a region.

Locators provide both discovery and load balancing services. You configure clients with a list of locator services and the locators maintain a dynamic list of member servers. By default, Geode clients and servers use port 40404 and multicast to discover each other.

Geode includes the following features:

- Combines redundancy, replication, and a "shared nothing" persistence architecture to deliver fail-safe reliability and performance.
- Horizontally scalable to thousands of cache members, with multiple cache topologies to meet different enterprise needs. The cache can be distributed across multiple computers.
- Asynchronous and synchronous cache update propagation.
- Delta propagation distributes only the difference between old and new versions of an object (delta) instead of the entire object, resulting in significant distribution cost savings.
- Reliable asynchronous event notifications and guaranteed message delivery through optimized, low latency distribution layer.
- Data awareness and real-time business intelligence. If data changes as you retrieve it, you see the changes immediately.
- Integration with Spring Framework to speed and simplify the development of scalable, transactional enterprise applications.
- JTA compliant transaction support.
- Cluster-wide configurations that can be persisted and exported to other clusters.
- Remote cluster management through HTTP.
- REST APIs for REST-enabled application development.
- Rolling upgrades may be possible, but they will be subject to any limitations imposed by new features.

Geode in 5 minutes

How to Get Apache Geode

You can download Apache Geode from the website, run a Docker image, or install with homebrew on OSX. Application developers can load dependencies from Maven Central.

Maven

```
<dependencies>

    <dependency>
        <groupId>org.apache.geode</groupId>
        <artifactId>geode-core</artifactId>
        <version>$VERSION</version>
    </dependency>
</dependencies>
```

Gradle

```
dependencies {
  compile "org.apache.geode:geode-core:$VERSION"
}
```

Start the cluster

Download Geode by following one of the methods described above, and follow the installation instructions at in the posted manual at http://geode.apache.org/docs/.

With a path that contains the `bin` directory of the installation, start a locator and server:

```
$ gfsh
```

```
gfsh> start locator
```

```
gfsh> start server
```

Create a region:

```
gfsh> create region --name=hello --type=REPLICATE
```

Write a client application (this example uses a Gradle build script):

src/main/java/HelloWorld.java

```
import java.util.Map;
import org.apache.geode.cache.Region;
import org.apache.geode.cache.client.*;

public class HelloWorld {
  public static void main(String[] args) throws Exception {
    ClientCache cache = new ClientCacheFactory()
      .addPoolLocator("localhost", 10334)
      .create();
    Region<String, String> region = cache
      .<String, String>createClientRegionFactory(ClientRegionShortcut.CACHING_PROXY)
      .create("hello");

    region.put("1", "Hello");
    region.put("2", "World");

    for (Map.Entry<String, String>  entry : region.entrySet()) {
      System.out.format("key = %s, value = %s\n", entry.getKey(), entry.getValue());
    }
    cache.close();
  }
}
```

Build and run the `HelloWorld` example.

```
$ gradle run
```

The application will connect to the running cluster, create a local cache, put some data in the cache, and print the cached data to the console:

```
key = 1, value = Hello
key = 2, value = World
```

Finally, shutdown the Geode server and locator:

```
gfsh> shutdown --include-locators=true
```

For more information see the Geode Examples repository or the documentation .

Application Development

Geode applications can be written in these client technologies:

a number of client technologies:

- Java client or peer
- Java using the Geode client API or embedded using the Geode peer API
- .Net and C++ via the Native Client
- REST
- Redis
- memcached

The following libraries are available external to the Apache Geode project:

- Spring Data Geode
- Spring Cache
- Python