BigMemory Go Installation Guide

Innovation Release

Version 4.3.5

April 2018

TERRACOTTA

This document applies to BigMemory Go Version 4.3.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: BMG-IG-435-20180417

Table of Contents

Installing and Configuring BigMemory Go..... 5
Installing BigMemory Go..... 6
Configuring BigMemory Go..... 7
Working with Terracotta License Files..... 9
Learn More about How BigMemory Go Works..... 10

1 Installing and Configuring BigMemory Go

- Installing BigMemory Go 6
- Configuring BigMemory Go 7
- Working with Terracotta License Files 9
- Learn More about How BigMemory Go Works 10

Installing BigMemory Go

Installing BigMemory Go is as easy as downloading the kit and ensuring that the correct files are on your application's classpath. The only platform requirement is using JDK 1.6 or higher.

To install BigMemory Go

1. If you do not have a BigMemory Go kit, download it from "http://terracotta.org/downloads/bigmemorygo".

   The kit is packaged as a tar.gz file. Unpack it on the command line or with the appropriate decompression application.

2. The following JARs are found in the kit's lib directory and must be added to your application's classpath:

   - ehcache-ee-<*version*> .jar

     This file contains the API to BigMemory Go.

   - slf4j-api-<*version*> .jar

     This file is the bridge, or logging facade, to the BigMemory Go logging framework.

3. Save the BigMemory Go license-key file to the BigMemory Go home directory. This file, called terracotta-license.key, was attached to an email you received after registering for the BigMemory Go download.

   Alternatively, you can add the license key file to your application's classpath, or explicitly specify the location of the license file as described in "Working with Terracotta License Files" on page 9.

4. BigMemory Go uses Ehcache as its user-facing interface. To configure BigMemory Go, create or update an Ehcache configuration file to specify how much off-heap in-memory storage your want to use. You may also configure BigMemory to write data to a local disk store for fast restart. For example:

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="http://ehcache.org/ehcache.xsd"
     name="myBigMemoryGoConfig">
  <!-- Tell BigMemory where to write its data to disk. -->
  <diskStore path="/path/to/my/disk/store/directory"/>
  <!-- set "maxBytesLocalOffHeap" to the amount of off-heap memory you
  want to use.  This memory is invisible to the Java garbage collector,
  providing gigabytes to terabytes of in-memory data without garbage
  collection pauses. -->
  <cache name="myBigMemoryGoStore"
        maxBytesLocalHeap="512M"
        maxBytesLocalOffHeap="32G">
    <!-- Tell BigMemory to use the "localRestartable" persistence
    strategy for fast restart (optional). -->
    <persistence strategy="localRestartable"/>
  </cache>
</ehcache>
```

Set maxBytesLocalOffHeap to the amount of off-heap storage you want to use. Depending on your data and how much physical RAM you have available, you can use just a few gigabytes to multiple terabytes of off-heap memory in a single JVM for ultra-fast access with no garbage collection.

Name this configuration file ehcache.xml and place it in the top-level of your classpath.

For more information on configuration options, see the *BigMemory Go Configuration Guide* and to the reference ehcache.xml configuration file in the config-samples directory of the BigMemory Go kit.

5. Use the -XX:MaxDirectMemorySize Java option to allocate enough direct memory in the JVM to accomodate the off-heap storage specified in your configuration, plus at least 250MB to allow for other direct memory usage that might occur in your application. For example:

   -Dcom.tc.productkey.path=/path/to/terracotta-license.key

   Set `MaxDirectMemorySize` to the amount of BigMemory you have. For more information about this step, see "Allocating Direct Memory in the JVM in the *BigMemory Go Configuration Guide*.

   Also, allocate at least enough heap using the `-Xmx` Java option to accommodate the on-heap storage specified in your configuration, plus enough extra heap to run the rest of your application. For example:

   ```
   Xmx1g
   ```

6. Refer to the in the download kit for examples of how to employ the various features and capabilities of BigMemory Go. For information about the code samples, see *About the BigMemory Go Code Samples* in the online documentation for BigMemory Go.

Configuring BigMemory Go

Automatic Resource Control

Automatic Resource Control (ARC) gives you fine-grained controls for tuning performance and enabling trade-offs between throughput, latency and data access. Independently adjustable configuration parameters include differentiated tier-based sizing and pinning hot or eternal data in the most effective tier.

Dynamically Sizing Stores

Tuning often involves sizing stores appropriately. There are a number of ways to size the different BigMemory Go data tiers using simple configuration sizing attributes. For information on how to tune tier sizing by configuring dynamic allocation of memory and automatic balancing, see "Sizing Storage Tiers" in the *BigMemory Go Configuration Guide*.

Pinning Data

One of the most important aspects of running an in-memory data store involves managing the life of the data in each BigMemory Go tier. For information on the pinning, expiration, and eviction of data, see "Managing Data Life" in the *BigMemory Go Configuration Guide*.

Fast Restartability - FRS

BigMemory Go has full fault tolerance, allowing for continuous access to in-memory data after a planned or unplanned shutdown, with the option to store a fully consistent record of the in-memory data on the local disk at all times. For information on data persistence, fast restartability, and using the local disk as a storage tier for in-memory data (both heap and off-heap stores), see "Configuring Fast Restart" in the *BigMemory Go Configuration Guide*.

Search

Search billions of entries - gigabytes or even terabytes of data - with results returned in less than a second. Data is indexed without significant overhead, and features like "GroupBy', direct support for handling null values, and optimization around handling huge results sets are included. The Search API provides the ability for data to be looked up based on multiple criteria instead of just keys. You can query BigMemory data using either simple SQL statements or the Search API. For more information, see "Searching a Cache" and "Searching with BigMemory SQL" in the *BigMemory Go Developer Guide*.

Transactional Caching

Transactional modes are a powerful extension for performing atomic operations on data stores, keeping your data in sync with your database. For background and configuration information for BigMemory Go transactional modes, see "Transaction Support" in the *BigMemory Go Developer Guide*. Explicit locking is another API that can be used as a custom alternative to XA Transactions or Local transactions (see "Using Explicit Locking" in the *BigMemory Go Developer Guide*).

Administration and Monitoring

The Terracotta Management Console (TMC) is a web-based monitoring and administration application for tuning cache usage, detecting errors, and providing an easy-to-use access point to integrate with production management systems. For more information, see the *Terracotta Management Console User's Guide*.

As an alternative to the TMC, standard JMX-based administration and monitoring is available. See the *BigMemory Go Operations Guide*.

For logging, BigMemory Go uses the flexible SLF4J logging framework. See the *BigMemory Go Operations Guide*.

Working with Terracotta License Files

A Terracotta license file is required to run enterprise versions of Terracotta products. The name of the file is terracotta-license.key and must not be changed. Trial versions of Terracotta enterprise products expire after a trial period. Expiration warnings are issued both to logs and standard output to allow enough time to contact Terracotta for an extension.

Each node using an enterprise version of Terracotta software requires a copy of the license file or configuration that specifies the file's location. By default, the file is provided in the root directory of the Terracotta software kit. To avoid having to explicitly specify the file's location, you can leave it in the kit's root directory.

Or, more generally (if you are using BigMemory Max), ensure that the resource / terracotta-license.key is on the same classpath as the standard Terracotta runtime JARs. For example, the license file could be placed in `WEB-INF/classes` when using a web application.

Explicitly Specifying the Location of the License File

If the file is in the Terracotta installation directory, you can specify it with:

```
-Dtc.install-root=/path/to/terracotta-install-dir
```

If the file is in a different location, you can specify it with:

```
-Dcom.tc.productkey.path=/path/to/terracotta-license.key
```

Alternatively, the path to the license file can be specified by adding the following to the beginning of the Terracotta configuration file (tc-config.xml by default):

```
<tc-properties>
    <property name="productkey.path" value="path/to/terracotta-license.key" />
    <!-- Other tc.properties here. -->
</tc-properties>
```

To refer to a license file that is in a WAR or JAR file, substitute `productkey.resource.path` for `productkey.path`.

Verifying Products and Features

There are a number of ways to verify what products and features are allowed and what limitations are imposed by your product key. The first is by looking at the readable file (`terracotta-license.key`) containing the product key.

Second, at startup Terracotta software logs a message detailing the product key. The message is printed to the log and to standard output. The message should appear similar to the following:

```
2010-11-03 15:56:53,701 INFO - Terracotta license loaded from
/Downloads/terracotta-ee-3.4.0/terracotta-license.key
Capabilities: DCV2, authentication, ehcache, ehcache monitor, ehcache offheap,
operator console, quartz, roots, server array offheap, server striping, sessions
Date of Issue: 2010-10-16
Edition: FX
```

```
Expiration Date: 2011-01-03
License Number: 0000
License Type: Trial
Licensee: Terracotta QA
Max Client Count: 100
Product: Enterprise Suite
ehcache.maxOffHeap: 200G
terracotta.serverArray.maxOffHeap: 200G
```

Learn More about How BigMemory Go Works

- Explore the code samples provided in the kit to learn what you can do with BigMemory Go. For information about the samples, see Code Samples in the online documentation for BigMemory Go.
- Review "Configuring BigMemory Go" in the *BigMemory Go Configuration Guide* to learn about how BigMemory Go is configured.
- Review "Configuring Storage Tiers" in the *BigMemory Go Configuration Guide* to learn about BigMemory Go architecture.