Table Of Contents

1. GridGain In-Memory Database

GridGain's In-Memory Database is distributed, object-based, ACID transactional, in-memory, SQL+NoSQL key-value database. Let's look at each characteristic individually:

Charactersitic	Description
Distributed	Unlike sharding systems - GridGain is distributed by default and has been built this way from the ground up. In other words, GridGain allows you to install itself on multiple computers and it will automatically coordinate data management across these computers seamlessly for the user so that the entire cluster can be viewed as a single in-memory database. Moreover, you can seamlessly add and remove nodes from topology and GridGain will properly handle the data repartitioning, transactions, indexing, etc. **NOTE:** GridGain has been independently tested and have production installations with over thousands computers in the grid.
SQL+NoSQL Key-Value	Data in in-memory database is kept in a form of `key/value` pairs. In a simplistic approximation you can think of in-memory database as a collection of distributed hash maps. Both `keys` and `values` can be of any type (including any containers or user-defined types). It is important to note that key/value representation doesn't mean primitive types or byte-arrays only. GridGain just as easily supports document APIs where value represent JSON or any other document-style representation. GridGain provides many interfaces to query and process its data: SQL, Key-Value, MapReduce, File System, MPP and MPI-like processing.
Transactional	Unlike most of NoSQL products that provide only **E**ventual **C**onsistency (EC) model, GridGain provides full ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability) compliant transactions that ensure guaranteed consistency. Moreover, GridGain provides various performance optimizations that allow users to relax some of the ACID properties in exchange for even greater performance.
In-Memory	GridGain treats memory (on-heap and off-heap) as a primary media to store application data. It also natively supports disk-based swap space to integrate with non-volatile storage like HDDs, SSDs, SANs, etc. as well as cache stores that allow GridGain in-memory database to read-through from and write-through to an existing data stores like SQL RDBMS, HDFS, HBase, etc. However, the main algorithms of GridGain (ACID transactional system, MapReduce, SQL processing, etc.) are **heavily optimized** for in-memory processing.
Object-Based	As we mentioned above - both keys and values can be of any type supported by JVM. That allows to store any business domain objects of any internal complexity in in-memory database directly without any need to marshall and demarshall from any external representations such as binary, XML or JSON protocols. This in turn provides two key advantages: 1) Developers can work directly with domain objects enjoying natural, convenient and simple to use APIs, and 2) Object-based storage leads to better performance as objects don't have to be constantly marshaled from

	one presentation to another to be used by the user or by the underlying system

Key Features

> **In-Memory Database** edition includes everything provided by **In-Memory HPC** edition.

GridGain In-Memory Database provides the following key features:

- Distributed java-based in-memory key-value store
- Distributed ANSI SQL and Lucene in-memory query capability over cached data
- PESSIMISTIC and OPTIMISTIC ACID distributed transactions
- Affinity routing with compute grid
- Local, fully replicable, and partitioned cache types
- Partitioned caches with active backups
- Support for data replication and invalidation in sync/async modes
- Concurrent, transactional, and delayed data preloading
- Pluggable expiration policies (LRU, LIRS, random, time-based)
- Read-through and write-through logic with pluggable cache store
- Synchronous and asynchronous cache operations
- MVCC-based concurrency with HyperLocking technology
- Pluggable network segmentation (split brains) resolution
- Pluggable data overflow storage
- READ*COMMITTED, REPEATABLE*READ, SERIALIZABLE isolation levels
- JTA/JCA transaction integration
- Write-behind cache store support
- Datacenter replication
- Customizable/pluggable data indexing support
- JDBC driver for in-memory object data store
- Off-Heap BigMemory support
- Tiered storage with on-heap, off-heap, swap space, SQL, and Hadoop
- Zero deployment for data
- OOP and FP-based APIs for Java and Scala

2. GridGain Installation

GridGain distribution comes in a ZIP file that simply needs to be unzipped and `GRIDGAIN_HOME` environment variable should be set to point to `distro` folder under GridGain installation.

You can use any tools to perform this installation on multiple computers. There are no additional steps required for GridGain installation in such multi machine setup.

Installation requirements:

1. Windows, Linux, or MacOS environment.
2. Java 7 (latest update is advisable).
3. Point `JAVA_HOME` environment variable to your JDK or JRE installation.
4. Point `GRIDGAIN_HOME` environment variable to the `distro` folder under GridGain installation.

2.1 Check GridGain Installation

To verify GridGain installation, you can execute the GridGain startup script.

The following command will startup GridGain with default configuration using Multicast node discovery.

```
bin/ggstart.{sh|bat}
```

The following command will startup GridGain with default configuration using TCP node discovery for all nodes running on local host.

```
bin/ggstart.{sh|bat} config/tcp/spring-tcp-vm.xml
```

If GridGain was installed successfully, the output from above commands should produce no exceptions or errors. Note that you may see some warnings during startup, but this is OK as they are meant to inform that certain functionality is turned on or off by default.

You can execute the above commands multiple times on the same machine and make sure that nodes discover each other. Here is an example of log printout when 2 nodes join topology:

```
... Topology snapshot [nodes=2, CPUs=8, hash=0xD551B245]
```

You can also start GridGain Management Console, called Visor, and observe started nodes show up on Visor Dashboard. To startup Visor in GUI mode, you should execute the following script:

```
/bin/ggvisorui.{sh|bat}
```

2.2 Running GridGain Examples

GridGain comes with many well documented examples. All examples have documentation about how they should be started and what the expected outcome should be. Start off by running simple examples, such as examples under `org.gridgain.examples.hpc.helloworld` package for HPC, `org.gridgain.examples.database.putget` package for database, `org.gridgain.examples.ggfs` package for GGFS, `ogr.gridgain.examples.streaming` package for streaming, and then move to others.

For affinity colocation with HPC edition (i.e. colocation of computations with data) you should take a look at examples in `org.gridgain.examples.database.affinity` package. You should also take a look at examples under `org.gridgain.examples.database.loaddata` to see how GridGain to see various ways of loading data into GridGain including `realtime` example which loads and queries data concurrently in real time.

If you are using Eclipse or IntelliJ IDEA you can play with examples using pre-configured projects that are shipped with GridGain.

2.2.1 IDEA Users

IDEA users should open IDEA project file called `idea_users_open_this_file.ipr` located under GridGain installation.

2.2.2 Eclipse Users

Eclipse users should follow instructions specified in `eclipse_users_read_here.txt` under GridGain installation.

2.2.3 Other IDEs

If you are using IDE other than Eclipse or IDEA, you can still view GridGain examples by creating a project manually in IDE of your choice. Make sure to include `gridgain-x.x.x.jar` file and and libraries located under `libs` folder as dependencies into your project.

2.3 Configure GridGain Node Discovery

When using TCP-based discovery make sure to update configuration for IP Finder with actual IP addresses like so:

```
<property name="discoverySpi">
    <bean class="org.gridgain.grid.spi.discovery.tcp.GridTcpDiscoverySpi">
        <property name="ipFinder">
            <bean class="org.gridgain.grid.spi.discovery.tcp.ipfinder.vm.GridTcpDiscoveryVmIpFinder">
                <property name="addresses">
                    <list>
                        <value>10.1.2.3:47500</value>
                        <value>10.1.2.4:47501</value>
                    </list>
                </property>
            </bean>
        </property>
    </bean>
</property>
```

On startup, GridGain node will try to connect to the specified IP addresses one-by-one until it succeeds.

> **NOTE:** you are only required to specify at least 1 IP address of the grid node that will be started first - other IP addresses are optional.

3. Starting Grid Nodes

Grid nodes can be started by executing `bin/ggstart.{sh|bat}` script and passing a relative path to GridGain configuration file. If no file is passed, then grid nodes are started with default configuration using Multicast discovery protocol.

Here is an example of how to start GridGain node with non-default configuration:

```
`bin/ggstart.sh config/tcp/spring-cache.xml`
```

Note that only TCP discovery is supported for In-Memory Database (Multicast discovery is not supported). This means that you cannot use GridGain In-Memory Database with default configuration and should always pass configuration file with cache configuration along with TCP discovery configuration. Example of such configuration can be found in `examples/config/tcp/spring-cache.xml` configuration file.

4. Management & Monitoring with Visor

GridGain comes with GUI and CLI (command) based DevOps Managements Consoles delivering advance set of management and monitoring capabilities. Visor GUI is based on a standalone Java application and CLI version is built on top of Scala REPL providing fully scriptable and customizable DevOps environment.

To start Visor GUI console, execute the following command:

```
`bin/ggvisorui.sh`
```

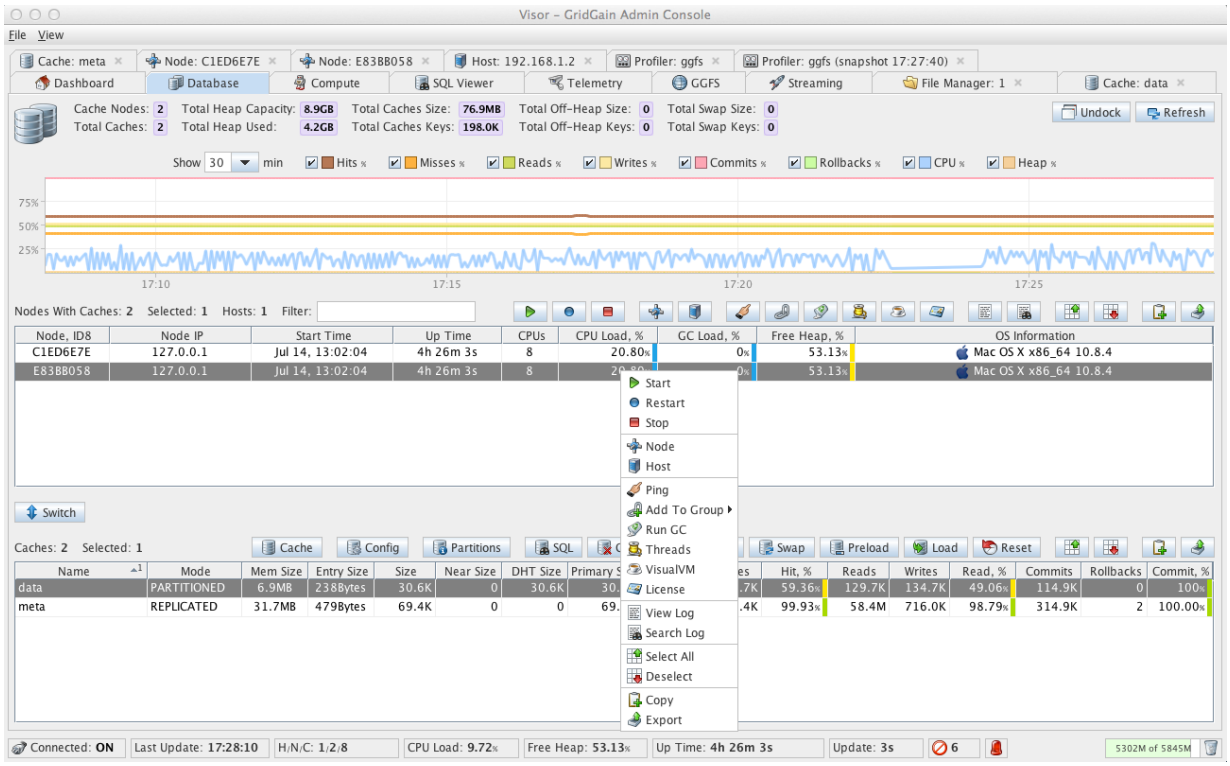
To start Visor in console mode you should execute the following command:

```
`bin/ggvisor.sh`
```

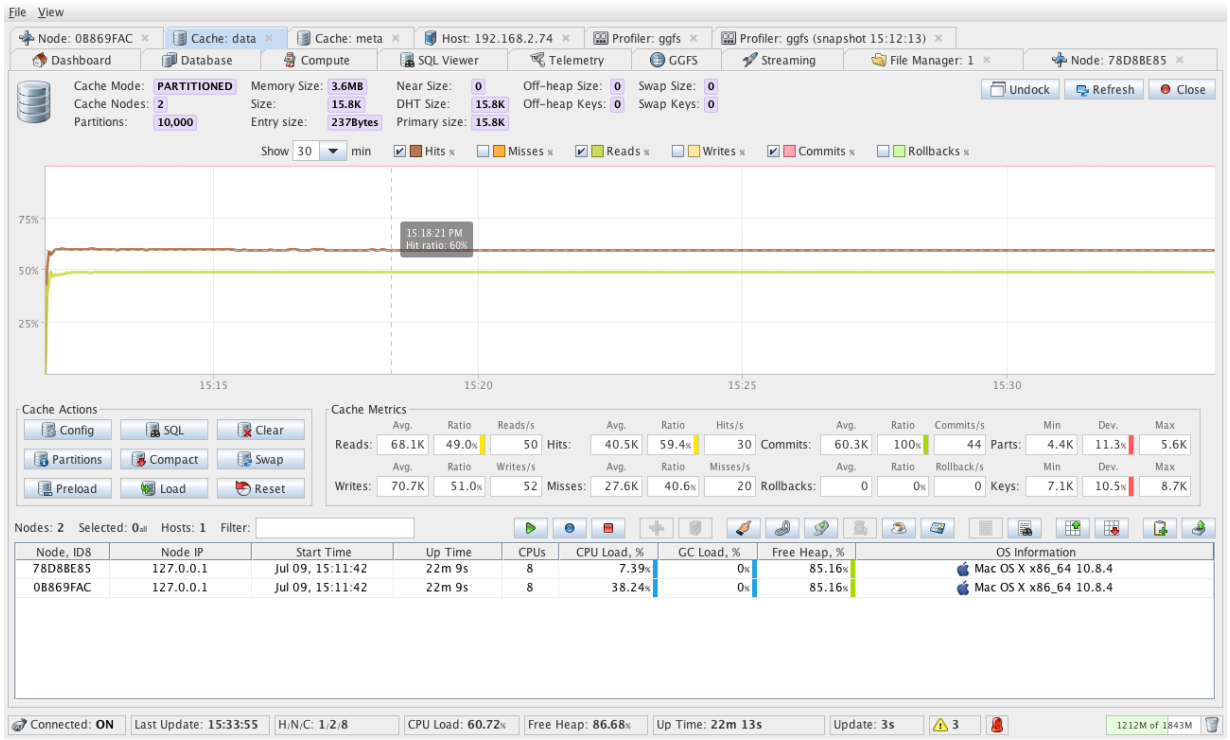
On Windows, run the same commands with `.bat` extension.

> **NOTE:** Visor GUI console has a much richer set of functionality over Visor command-based console. You should always prefer Visor GUI console whenever possible.

Here is an example of `Visor Database Tab` which provides overall view on database.



Here is an example of `Visor Cache Tab` which provides view on individual cache.



5. Scala Integration

GridGain provides a very nice and easy to use DSL for Scala users called `Scalar`. If you like Scala, take a look at Scalar examples located under `examples/scalar` folder.

6. Javadoc & Scaladoc

We spend significant amount of time on writing and maintaining Javadoc and Scaladoc for our product. All documentation is shipped with it and you can find it under `docs/javadoc` and `docs/scaladoc` sub-folder respectively.