FAQ

When should I use Geode?

Application developers and IT architects who need extremely fast processing and consistent data using open source software often run into trouble. When their applications are required to support thousands of concurrent transactions that access hundreds of gigabytes of operational data, they start having performance problems, or problems with the integrity of data.

Geode is an in-memory distributed database designed to provide high performance, low latency, extreme scale-out concurrency and consistency for data storage.

Unlike traditional relational databases with scaling limitations, Geode scales out horizontally across many nodes to provide low latency response for thousands of concurrent read and write operations on terabytes of data in memory.

Unlike many in-memory data grids, Geode can maintain a high degree of data consistency across many concurrent transactions and can operate as a highly available, resilient service. This makes it possible for users to deploy mission critical applications at very high scale.

Is Geode a mature technology?

Yes, Geode is an extremely mature and robust product that can trace its legacy all the way back to one of the first Object Databases for Smalltalk: GemStone. Geode (as GemFire™) was first deployed in the financial sector as the transactional, low-latency data engine used by multiple Wall Street trading platforms. Today Geode is used by over 600 enterprise customers for high-scale, 24x7 business critical applications. An example deployment includes China National Railways that uses Geode to run railway ticketing for the entire country of China with a 10 node cluster that manages 2 terabytes of "hot data" in memory, and 10 backup nodes for high availability and elastic scale.

How far can Geode scale?

Geode has been deployed to run mission-critical applications on clusters with more than 100 members managing terabytes of in-memory data. Adding capacity is as simple as spinning up a new node. Geode automatically configures the new member and reassigns data and loading across the cluster.

What platforms run Geode?

Geode runs on any platform with a 1.8 or more recent version JDK.

How does my application connect to a Geode cluster?

Geode provides Java client APIs that can be used by any other language running in the JVM (Scala, Groovy, Javascript, etc). A C++ library, .NET client, and a REST interface are also available.

The client drivers can be configured to cache data locally for improved performance. In addition, the driver provides single-hop network reads and writes for optimal performance.

Does Geode support zero downtime operation?

Yes, Geode provides rolling upgrade support, so a cluster can remain online even while being upgraded. In addition, Geode undergoes strict backwards compatibility testing to ensure that existing applications will continue to function. PDX serialization support for forwards and backwards data versioning allows a data model to seamlessly evolve.

Does Geode support transactions?

Yes, Geode provides atomic transactions (all data operations succeed or fail together). Transactions are executed on a single node to avoid expensive distributed lock operations. When the transaction is committed the results are replicated to the other cluster members. Within the context of a transaction updates will not be seen until they are committed. Data used in a transaction must be colocated.

What happens if a member runs out of memory?

Geode works to prevent resource issues by supporting LRU (least recently used) eviction. Eviction can be configured to overflow an entry to disk or remove it altogether. Expiration is also supported. The Resource Manager can be configured to generate alerts at eviction and critical memory usage thresholds. When a member is in a critical state further writes are blocked to allow the GC and eviction activities to restore the member to normal operation.

What happens if a node fails?

Geode provides data redundancy to ensure zero data loss when a node fails. In addition, availability zones can be configured to ensure that redundant data copies are hosted on different racks. Because Geode guarantees data consistency, the failover to the redundant copies is seamless.

If I shut down all members in a cache, will I lose data?

You can configure a Geode region to store its data to local disk. When the cluster is restarted, all members restore the in-memory data from disk. Keys are recovered first and values are recovered asynchronously. Members ensure consistency by exchanging version information.

How does Geode ensure strong consistency?

Cache updates are synchronously replicated to ensure consistency prior to acknowledging success. Concurrent modifications are handled through version stamps on cache entries.

Bulk synchronization is performed in the event of the failure of a member or when a member is restarted and is recovering from disk. Bulk synchronization is performed through interchange of version vectors with "exceptions".

How does Geode partition data?

Keys are hash-partitioned over a fixed number of buckets (the default bucket count is 113). Buckets are automatically balanced across the cluster members based on data size, redundancy, and availability zones.

How does Geode handle a network partition?

The network partition detection system is based on quorum and liveness checks. If a member shuts down due to a network partition it will periodically attempt to reconnect and rebuild its cache, recovering data either from disk or from redundant storage in other members.

Does Geode support JSR-107?

While Geode does not directly support JSR-107 (JCache) with an API implementation, it does provide indirect support via Spring. Spring Data Geode's Caching feature and support for Geode is built on the core Spring Framework's Cache Abstraction, which added support for JCache annotations in 4.1. Spring gives Geode developers the best part of JCache without requiring unnecessary or invasive coding patterns.

How can I contribute?

Please check the How to Contribute page.