Distributed caching using Hazelcast

By
Team High Calibre

Kulkarni, Anuvinda
Murthy, Megha
Dattatreya, Shweta

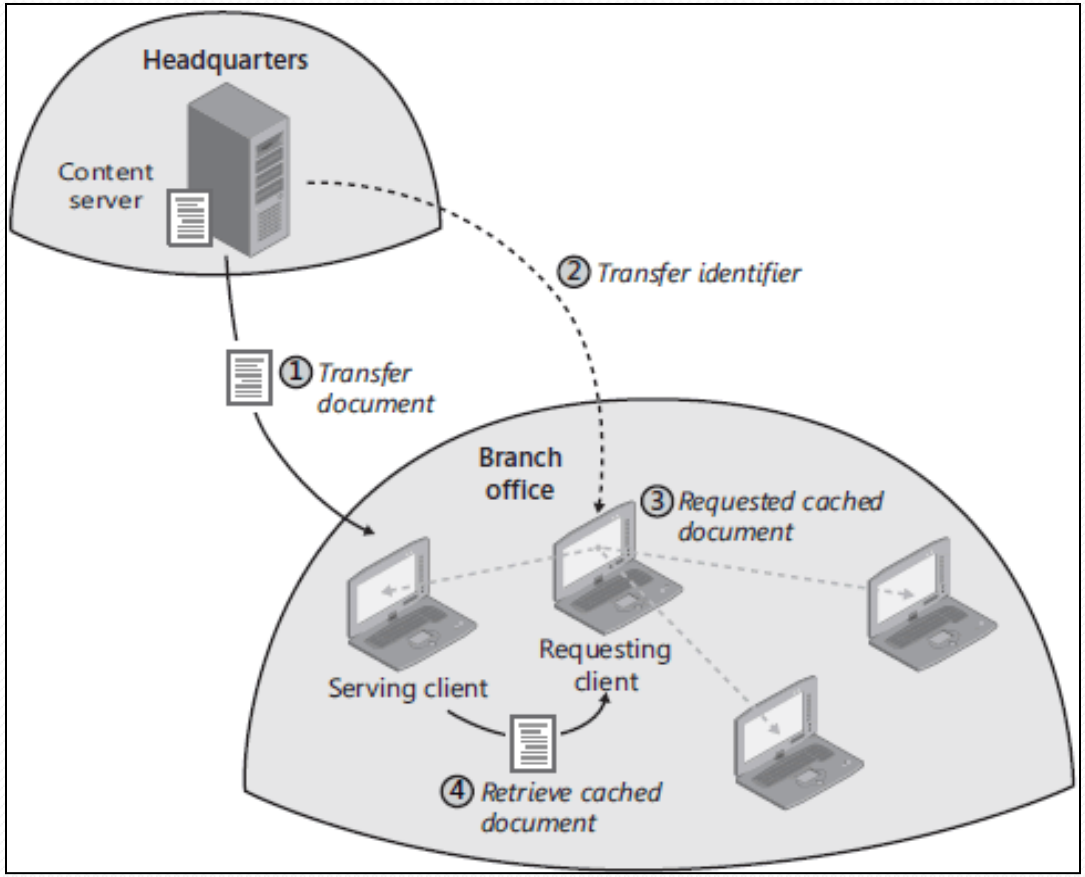
Agenda

- Problem Statement
- Solution to the problem statement using hazelcast
- Distributed Caching
- Technologies Used
  - Hazelcast
  - Hibernate
  - Hotwire API
- Basic workflow of the proposed system
- Execution and performance Evaluation
- Conclusion
- References

Problem Statement

- **Airline Use Case** - A passenger searches for flights by specifying the source, destination, and flight date and receives the list of flights and hotel deals for his destination.
- **Problem -** The search keeps hitting the Flight database very frequently. Database is overloaded and too slow. 80% of the searches are read-only. These read-only transactions keep hitting the database frequently thus making the response time slow.
- **Our Solution**
  - To reduce the load on database, deploy a distributed cache with several nodes running in the cluster
  - Cache data from database
  - Cached data is distributed equally between all the nodes
  - To avoid cache from ballooning, keep expiry on items.
  - Old untouched flight searches with hotel deals will expire from cache, but master data is always present in database and the hotel deals API

Distributed Caching



[Source: http://sourcedaddy.com/windows-7/how-distributed-cache-works.html]

Advantages of Distributed Caching

- High performance
- High scalability
- Reduced latency
- No single point of failure
- Session data is preserved
- Maintenance is easy
- Low cost

Technologies Used

- Hazelcast
- Hibernate
- Hotwire API
- MySql DB
- Java XPath API

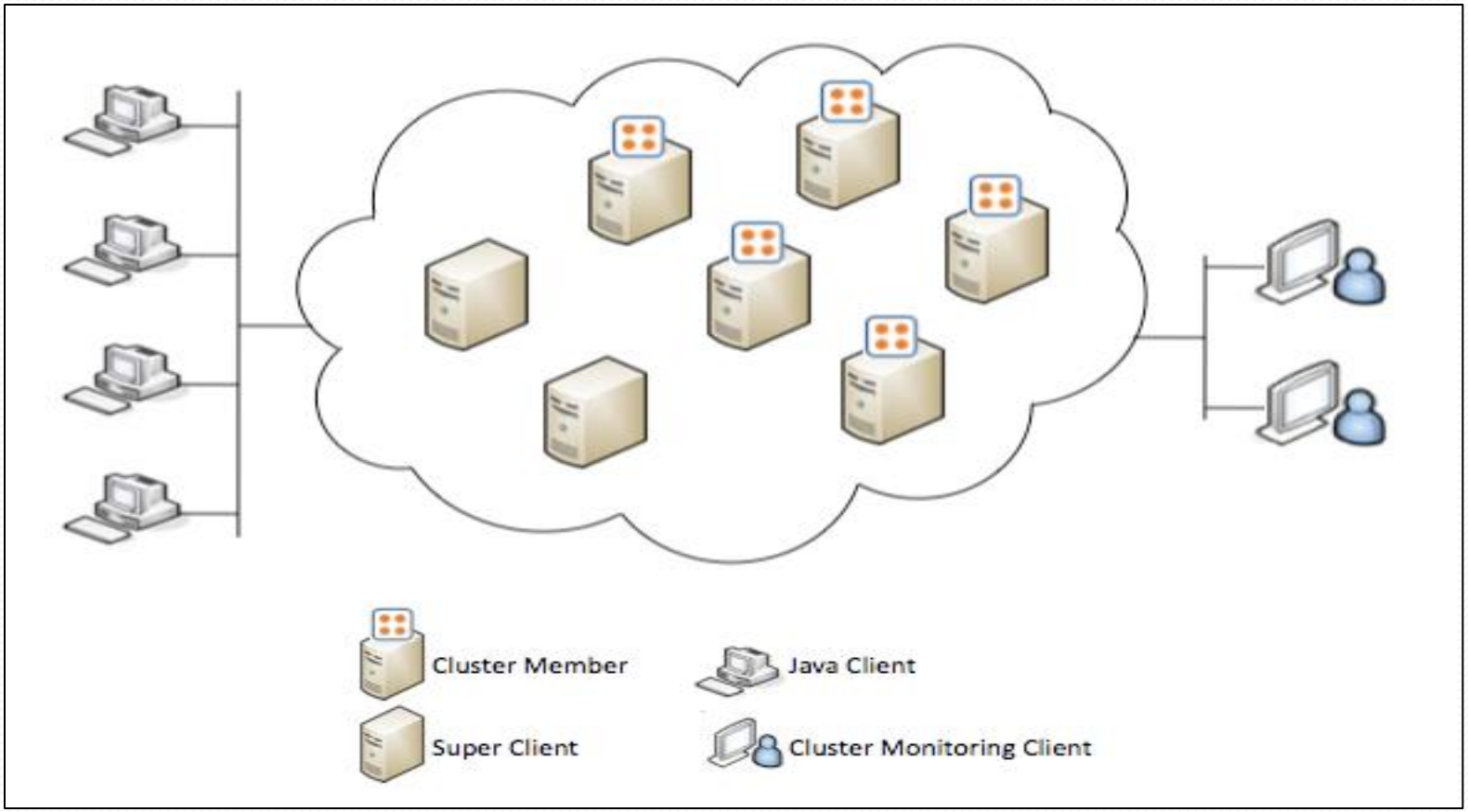
Hazelcast- A brief history

- Start-up founded in 2008
- By founders - Talip Ozturk, Fuad Malikov
- Open Source product under Apache License

What is Hazelcast?

- Clustering and scalable data distribution platform for java
- In-memory data grid

Hazelcast architecture



[Source: http://www.hazelcast.com/documentation.jsp]

Hazelcast features

- Distributed java.util.{Queue, Set, List, Map}
- Distributed java.util.concurrency.locks.Lock
- Distributed java.util.concurrent.ExecutorService

- Distributed MultiMap for one to many mapping
- Distributed Topic for publish/subscribe messaging
- Distributed Indexing and Query support
- Transaction support and J2EE container integration via JCA

- Socket level encryption for secure clustersWrite-Through and Write-Behind persistence for maps
- Java Client for accessing the cluster remotely
- Dynamic HTTP session clustering

- Support for cluster info and membership events
- Dynamic discovery, scaling, partitioning with backups, fail-over
- Web-based cluster monitoring

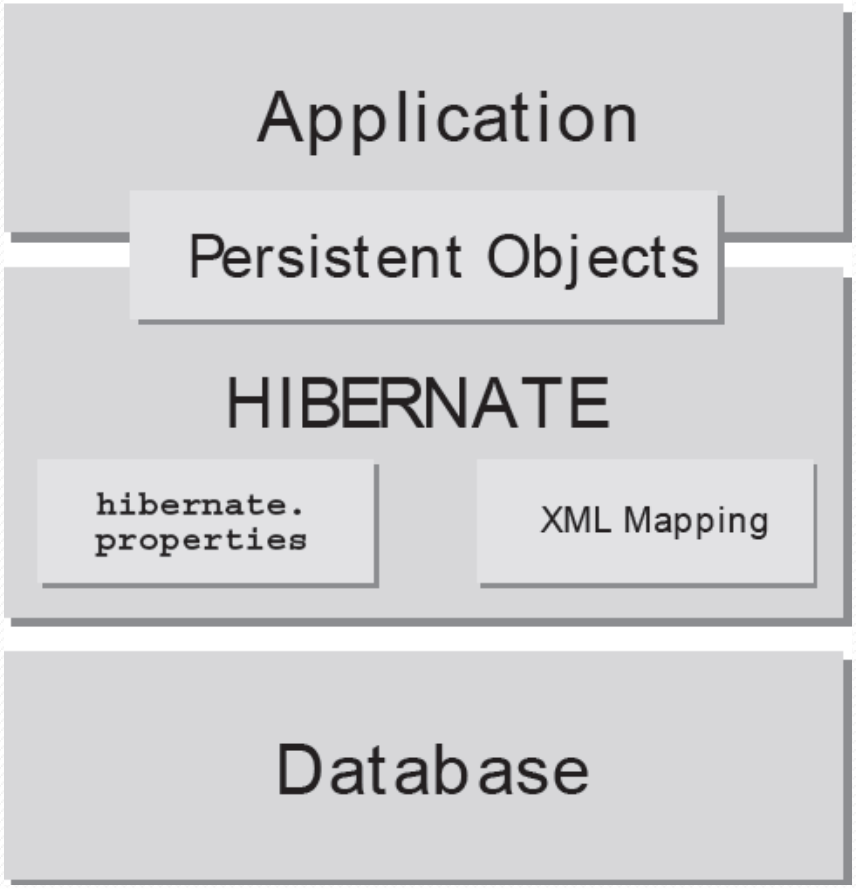
How does hazelcast help?

- Auto discovery of members in the cluster
- Fault tolerant
- Redistributing of data among all nodes even upon the entry of new node.

Hibernate

- An object-relational mapping (ORM) library for the Java language
- Hibernate is free software that is distributed under the GNU Lesser General Public License
- Primary feature is to map Java classes to database tables (and from Java data types to SQL data types)
- Mapping Java classes to database tables is accomplished through the configuration of an XML

Hibernate Architecture



[Source: http://hibernate.org/docs-hib-architecture

Hibernate Configuration File

- hibernate.cfg.xml
  - JDBC Driver class to use
  - Connection to the db
  - Connection Pool details
  - Second level of caching
- fight_details.hbm.xml
  - Mapping between the DB tables and Java classes

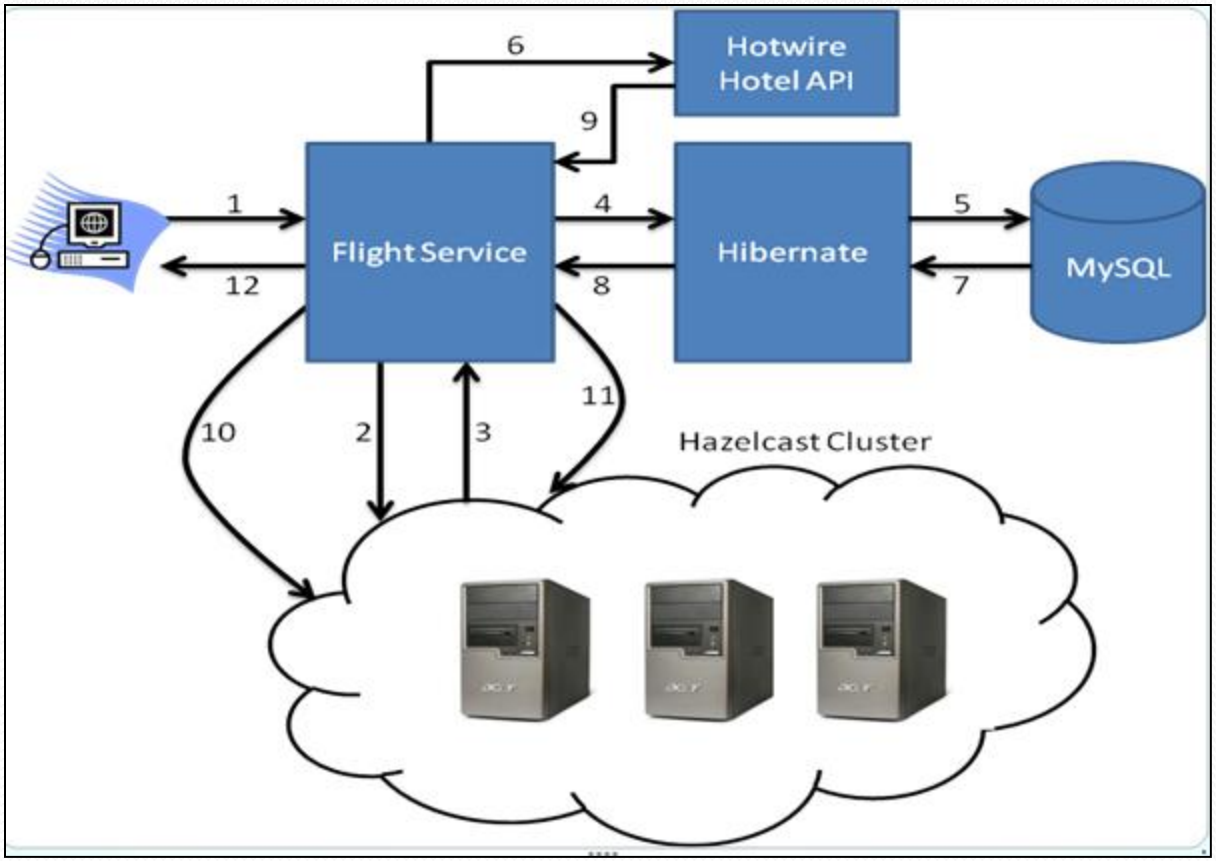
How does hibernate help?

- Greatly reduces complexity
- Easy configuration
- Connection pool

Hotwire API

- Hotel deals on hotwire
- Search result based on destination location
- Allows search based on multiple parameters
  - location
  - price
  - hotel star rating
  - travel dates
  - length of stay
  - restrict to weekend stay
  - time since deal was discovered

Basic Workflow of our model



Execution & Performance Evaluation

1. Run the CacheEngine to create cluster members
2. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-29”. Set the pre-fetch to true
3. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-29”. Set the pre-fetch to false
4. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-30”. Set the pre-fetch to false
5. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-28”. Set the pre-fetch to false
6. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-29”. Set the pre-fetch to false
7. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
8. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
9. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
10. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
11. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false

Execution & Performance Evaluation…continued

1. Run the CacheEngine to create cluster members
2. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-29”. Set the pre-fetch to true

   MySQL query - completed in [2768] milliseconds

   Hotwire API query - completed in [2338] milliseconds
3. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-29”. Set the pre-fetch to false

   Flight query from cache - completed in [2] milliseconds

   Hotel deals from cache - completed in [3] milliseconds
4. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-30”. Set the pre-fetch to false

   Flight query from cache - completed in [2] milliseconds

   Hotel deals from cache - completed in [2] milliseconds
5. Origin = “YUM”, destination = “LAX”, flightDate = “2011-06-28”. Set the pre-fetch to false

   Flight query from cache - completed in [2] milliseconds

   Hotel deals from cache - completed in [2] milliseconds
6. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-29”. Set the pre-fetch to false

   MySQL query - completed in [125] milliseconds

   Hotwire API query - completed in [1119] milliseconds

Execution & Performance Evaluation…continued

7. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
   MySQL query - completed in [176] milliseconds
   Hotel deals from cache - completed in [3] milliseconds
8. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
   Flight query from cache - completed in [2] milliseconds
   Hotel deals from cache - completed in [2] milliseconds
9. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
   Flight query from cache - completed in [3] milliseconds
   Hotel deals from cache - completed in [3] milliseconds
10. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
    MySQL query - completed in [159] milliseconds
    Hotwire API query - completed in [529] milliseconds
11. Origin = “SJC”, destination = “JFK”, flightDate = “2011-06-30”. Set the pre-fetch to false
    Flight query from cache - completed in [2] milliseconds
    Hotel deals from cache - completed in [2] milliseconds

Conclusion

- Hazelcast as an in-memory data grid - distributes data across cheap, commodity hardware with an open-source infrastructure
- Facilitates failover and scalability
- Disadvantage - technically not feasible to query using order by, group by or database joins in a distributed caching infrastructure
- Well-suited for applications that query using simple SQL-predicates
- Open source - easy to code

References

- http://code.google.com/edu/parallel/dsd-tutorial.html
- http://net.pku.edu.cn/~course/cs501/2011/resource/2006-Book-distributed%20systems%20principles%20and%20paradigms%202nd%20edition.pdf
- http://developer.hotwire.com/apps/mykeys
- http://developer.hotwire.com/docs/read/Hotel_Deals_API
- http://api.hotwire.com/v1/deal/hotel?apikey=q9w8hq5ecs4ag7gfcyn7g78a&limit=5&dest=NYC&distance=*~30&starrating=4~*&sort=price
- http://hc.apache.org/httpcomponents-client-ga/examples.html
- http://www.ibm.com/developerworks/library/x-javaxpathapi/index.html
- www.data.gov