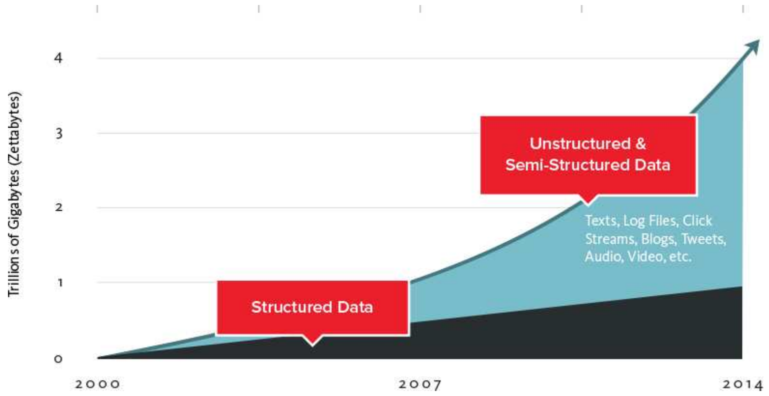
Infinispan

Vojtěch Juránek

JBoss - a division by Red Hat

11. 10. 2015, ACM RACS, Prague

Data today

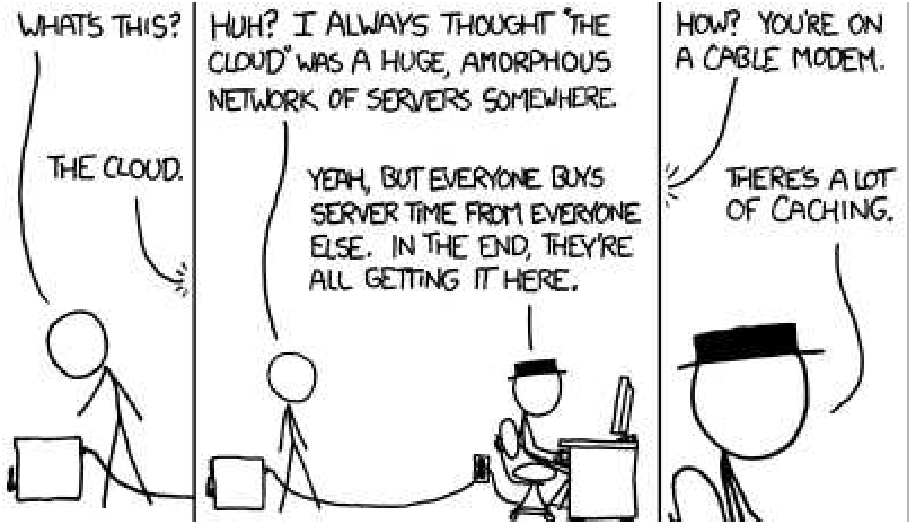


Source: http://www.couchbase.com/nosql-resources/what-is-no-sql

Data today

- Huge amount of data
  - You can scale up, but sooner or later you'll need to scale out
  - Need for highly scaleable solution also because of cost effectiveness
- Cloud achitecture - everything is ephemeral
- Lots of data is semi-structured or not structured at all (texts, log files, audio, video, click streams . . .)
  - Needed ability to store unstructured data (often for performance reasons)
  - But ability to talk to RDBMS is often needed as well

Why caching


WHAT'S THIS?
THE CLOUD.
HUH? I ALWAYS THOUGHT "THE CLOUD" WAS A HUGE, AMORPHOUS NETWORK OF SERVERS SOMEWHERE.
YEAH, BUT EVERYONE BUYS SERVER TIME FROM EVERYONE ELSE. IN THE END, THEY'RE ALL GETTING IT HERE.
HOW? YOU'RE ON A CABLE MODEM.
THERE'S A LOT OF CACHING.

Source: Part of xkcd #908

Why in-memory

- Lots of data is needed in real-time (BigData $\rightarrow$ FastData)
- Some tasks can be completed much faster when data are kept in memory
- Keeping data in memory during processing of whole application stack, not only during processing in one application in the stack
- With data replication you can keep your data only in memory (no need to store them in persistent storage)

Infinispan

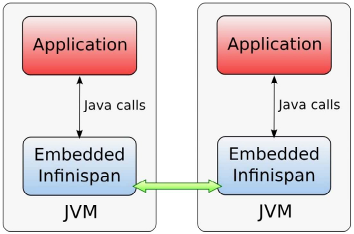
- Data grid platform, written in Java
- In-memory, (optionally) schema-less, No-SQL key-value data store
- Distributed cache - offers massive memory
- Elastic and scalable - can run on hundreds of nodes
- Highly available - no SPOF, resilient to node failures
- Concurrent (MVCC)
- Transactional
- Queryable
- Processing for streaming data

Infinispan

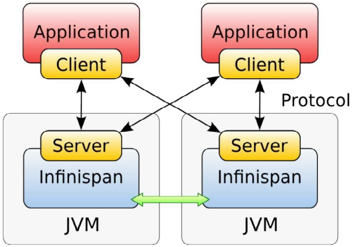
- 100% open-source
  - `https://github.com/infinispan`
  - Apache License, v2.0
- Member of couple research projects (funded by the EU)

Infinispan modes

- Embedded (library, in-VM)



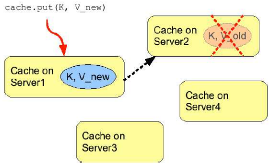
- Client-server (remote)



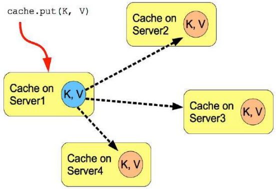
Clustering modes

Under the hood leverages JGroups project for clustering.

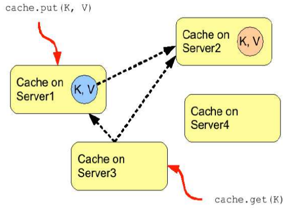
- Local - no clustering
- Invalidation



- Replicated

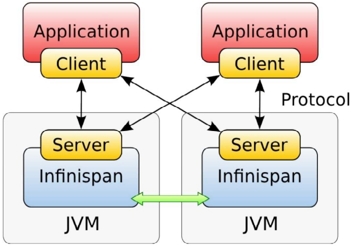


- Distributed



Remote protocols

- Hot Rod
  - hashing and topology aware
  - failover during topology changes
  - smart request routing
- Memcached
- REST



Hot Rod clients

Compatible with Java and non-Java platforms. Based on Protocol Buffers - Google's data interchange format.

Clients for

- Java
- C#
- C++
- Python
- Ruby

Python and Ruby clients have only basic functionality.

Cache stores

A way how to store cache content in some external (persistent) storage.
Two modes:

- Synchronous (write-through)
- Asynchronous (write-behind)

Cache stores:

- Single file store and soft-index file store
- JDBC and JPA cache stores
- LevelDB cache store
- Cloud cache store
- Remote store
- . . . and others

Also possible to define custom cache store.

Transactions

- JTA-compliant transactions
- Ensures consistency of data
- Optimistic and pessimistic locking available
- Read committed and repeatable read isolation levels
- Deadlock detection and recovery
- Data versioning

Querying

- Needs some data schema (protobuf file or annotations)
- Search for data using data attributes instead of keys item Featured attributes include keyword, range, fuzzy, wildcard, and phrase queries.
- Combine queries and aggregation functions (but doesn't support joins)
- Sort, filter, and paginate query results
- Support for index or non-indexed queries
- Lucene query API or fluent DSL API

Security

- Role based access control
- User authentication
- Node authentication and authorization
- Encryption of communication
- Audit logging
- Integration with LDAP and/or Kerberos server (includes Active Directory)

Some other features - brief and selective list

- Data eviction
- Full JSR-107 support (Java Temporary Caching API)
- CDI support
- Remote events
- Client near cache
- Rolling upgrades
- Cross data center replication (also Hot Rod clients support failover to another data center)
- Command line interface
- Map-reduce framework and distributed executors

Recently implemented features: Infinispan 8

- Functional API
- Distributed streaming
- Continuous querying, grouping and aggregation
- New management console
- Integration with Apache Spark and Hadoop
- . . . and more

Demo

Integration with Apache Spark

Some projects using Infinispan

- WildFly
- Hibernate
- Apache Camel
- Apache Marmotta
- CapeDwarf
- Immutant
- apiman
- . . . and others

http://infinispan.org/

Thank you for your attention!

Questions?