Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase vs Couchbase vs OrientDB vs Aerospike vs Neo4j vs Hypertable vs ElasticSearch vs Accumulo vs VoltDB vs Scalaris comparison

(Yes it's a long title, since people kept asking me to write about this and that too :) I do when it has a point.)

While SQL databases are insanely useful tools, their monopoly in the last decades is coming to an end. And it's just time: I can't even count the things that were forced into relational databases, but never really fitted them. (That being said, relational databases will always be the best for the stuff that has *relations*.)

NoSQL数据库之间的差别非常大，软件架构师的主要职责是在起初就要为项目选择一款合适的NoSQL。

But, the differences between NoSQL databases are much bigger than ever was between one SQL database and another. This means that it is a bigger responsibility on software architects to choose the appropriate one for a project right at the beginning.

In this light, here is a comparison of Cassandra, Mongodb, CouchDB, Redis, Riak, Couchbase (ex-Membase), Hypertable, ElasticSearch, Accumulo, VoltDB, Kyoto Tycoon, Scalaris, OrientDB, Aerospike, Neo4j and HBase:

The most popular ones

最流行

Redis (V3.0RC)

- **Written in:** C 磁盘备份的内存数据库
- **Main point:** Blazing fast 惊人的快
- **License:** BSD
- **Protocol:** Telnet-like, binary safe 二进制安全
- Disk-backed in-memory database,
- Dataset size limited to computer RAM (but can span multiple machines' RAM with clustering) 数据集大小限制于机器的RAM 可以使用集群来跨越多台机器的RAM
- Master-slave replication, automatic failover 主-从复制，自动故障转移
- Simple values or data structures by keys 丰富的数据结构支持

Cassandra (2.0)

- **Written in:** Java
- **Main point:** Store *huge* datasets in "almost" SQL SQL中的海量数据存储
- **License:** Apache
- **Protocol:** CQL3 & Thrift CQL3与SQL非常相似，但受到可伸缩性的一些限制（没有 联合、聚合 函数）
- CQL3 is very similar SQL, but with some limitations that come from the scalability (most notably: no JOINs, no aggregate functions.)
- CQL3 is now the official interface. Don't look at Thrift, unless you're working on a legacy app. This way, you can live

- but complex operations like ZREVRANGEBYSCORE.
- INCR & co (good for rate limiting or statistics) INCR：速率限制、统计
- Bit operations (for example to implement bloom filters)
- Has sets (also union/diff/inter)
- Has lists (also a queue; blocking pop) 列表：队列
- Has hashes (objects of multiple fields)
- Sorted sets (high score table, good for range queries) 有序集合：排行榜、范围查询
- Lua scripting capabilities (!)
- Has transactions (!)
- Values can be set to expire (as in a cache) 值可以被设置过期时间(缓存)
- Pub/Sub lets one implement messaging

Best used: For rapidly changing data with a foreseeable database size (should fit mostly in memory). 快速改变的可预见数据集大小的数据(满足大部分在内存中)

For example: To store real-time stock prices. Real-time analytics. Leaderboards. Real-time communication. And wherever you used memcached before. 实时的股票价格、实时的分析、排行榜、实时的通讯、先前使用Memcached的地方

MongoDB (2.6.7)

- **Written in:** C++ 保留一些友好的SQL特性(查询、索引)
- **Main point:** Retains some friendly properties of SQL. (Query, index)
- **License:** AGPL (Drivers: Apache)
- **Protocol:** Custom, binary (BSON)
- Master/slave replication (auto failover with replica sets) 主/从复制(副本集的自动故障转移)
- Sharding built-in 内建分片功能
- Queries are javascript expressions
- Run arbitrary javascript functions server-side 数据更新方式为直接修改
- Better update-in-place than CouchDB
- Uses memory mapped files for data storage 使用内存映射文件进行数据存储
- Performance over features 性能优于特性
- Journaling (with --journal) is best turned on 日志记录最好打开
- On 32bit systems, limited to ~2.5Gb

without understanding ColumnFamilies, SuperColumns, etc.

- Querying by key, or key range 查询：键、键范围
- (secondary indices are also available)
- Tunable trade-offs for distribution and replication (N, R, W) 分布和复制的权衡
- Data can have expiration (set on INSERT). 写入要快于读取(当读取是磁盘密集型时)
- Writes can be much faster than reads (when reads are disk-bound)
- Map/reduce possible with Apache Hadoop
- All nodes are similar, as opposed to Hadoop/HBase 所有节点都是相似的
- Very good and reliable cross-datacenter replication 很好的、可靠的跨数据中心复制
- Distributed counter datatype. 分布式的计数器数据类型
- You can write triggers in Java.

Best used: When you need to store data so huge that it doesn't fit on server, but still want a friendly familiar interface to it. 需要存储的数据巨大

For example: Web analytics, to count hits by hour, by browser, by IP, etc. Transaction logging. Data collection from huge sensor arrays. Web分析、按小时、浏览器、IP统计命中率 事务日志、收集于巨大的传感器阵列的数据集

ElasticSearch (0.20.1)

- **Written in:** Java
- **Main point:** Advanced Search 高级搜索
- **License:** Apache
- **Protocol:** JSON over HTTP (Plugins: Thrift, memcached)
- Stores JSON documents 存储JSON文档
- Has versioning 版本控制
- Parent and children documents
- Documents can time out
- Very versatile and sophisticated querying, scriptable 非常灵活和复杂的查询
- Write consistency: one, quorum or all 写一致性
- Sorting by score (!) 按分数排序
- Geo distance sorting

- Text search integrated 集成文本搜索
- GridFS to store big data + metadata (not actually an FS)
- Has geospatial indexing
- Data center aware 数据中心感知

Best used: If you need dynamic queries. If you prefer to define indexes, not map/reduce functions. If you need good performance on a big DB. If you wanted CouchDB, but your data changes too much, filling up disks.

For example: For most things that you would do with MySQL or PostgreSQL, but having predefined columns really holds you back.

动态查询
定义索引
在大数据集上需要更好的性能
拥有CouchDB的特性，但数据变化太多而填满磁盘

- Fuzzy searches (approximate date, etc) (!)
- Asynchronous replication 异步复制
- Atomic, scripted updates (good for counters, etc) 原子脚本更新
- Can maintain automatic "stats groups" (good for debugging)
- Still depends very much on only one developer (kimchy). 社区不活跃

Best used: When you have objects with (flexible) fields, and you need "advanced search" functionality. 对象的字段需要高级的搜索功能

For example: A dating service that handles age difference, geographic location, tastes and dislikes, etc. Or a leaderboard system that depends on many variables.

一个约会服务
一个依赖很多维度的排行榜系统

Classic document and BigTable stores

经典的文档和大数据存储

CouchDB (V1.2)

- **Written in:** Erlang
- **Main point:** DB consistency, ease of use 数据库一致性、易于使用
- **License:** Apache
- **Protocol:** HTTP/REST
- Bi-directional (!) replication, 双向复制
- continuous or ad-hoc,
- with conflict detection,
- thus, master-master replication. (!) 双主复制
- MVCC - write operations do not block reads MVCC-写入操作不会阻塞读取
- Previous versions of documents are available
- Crash-only (reliable) design
- Needs compacting from time to time
- Views: embedded map/reduce
- Formatting views: lists & shows
- Server-side document validation possible
- Authentication possible
- Real-time updates via '_changes' (!)

HBase (V0.92.0)

- **Written in:** Java
- **Main point:** Billions of rows X millions of columns 十亿行 * 百万列
- **License:** Apache
- **Protocol:** HTTP/REST (also Thrift)
- Modeled after Google's BigTable
- Uses Hadoop's HDFS as storage 使用HDFS作为存储
- Map/reduce with Hadoop
- Query predicate push down via server side scan and get filters
- Optimizations for real time queries 实时查询优化
- A high performance Thrift gateway 高性能的Thrift网关
- HTTP supports XML, Protobuf, and binary HTTP支持Protobuf
- Jruby-based (JIRB) shell
- Rolling restart for configuration changes and minor upgrades
- Random access performance is like MySQL 随机访问性能 像MySQL
- A cluster consists of several different types of nodes

- Attachment handling
- thus, CouchApps (standalone js apps)

Best used: For accumulating, occasionally changing data, on which pre-defined queries are to be run. Places where versioning is important.

For example: CRM, CMS systems. Master-master replication is an especially interesting feature, allowing easy multi-site deployments.

Accumulo (1.4)

- **Written in:** Java and C++
- **Main point:** A BigTable with Cell-level security 一个单元级安全的BigTable
- **License:** Apache
- **Protocol:** Thrift
- Another BigTable clone, also runs of top of Hadoop BigTable克隆版，运行在Hadoop之上
- Originally from the NSA
- Cell-level security 单元级安全
- Bigger rows than memory are allowed
- Keeps a memory map outside Java, in C++ STL
- Map/reduce using Hadoop's facitlities (ZooKeeper & co)
- Some server-side programming

Best used: If you need to restict access on the cell level. 需要在单元级上严格访问

For example: Same as HBase, since it's basically a replacement: Search engines. Analysing log data. Any place where scanning huge, two-dimensional join-less tables are a requirement.

Best used: Hadoop is probably still the best way to run Map/Reduce jobs on huge datasets. Best if you use the Hadoop/HDFS stack already. 随机实时的读、写操作 (已经在使用Hadoop/HDFS系列)

For example: Search engines. Analysing log data. Any place where scanning huge, two-dimensional join-less tables are a requirement. 搜索引擎 Facebook 消息数据库 扫描巨大的、二维无联合的表

Hypertable (0.9.6.5)

- **Written in:** C++ 更快、更小的HBase
- **Main point:** A faster, smaller HBase
- **License:** GPL 2.0
- **Protocol:** Thrift, C++ library, or HQL shell
- Implements Google's BigTable design
- Run on Hadoop's HDFS
- Uses its own, "SQL-like" language, HQL
- Can search by key, by cell, or for values in column families. 通过 键、单元、列簇值 搜索
- Search can be limited to key/column ranges. 搜索可以限制于 键/列 范围
- Sponsored by Baidu
- Retains the last N historical values
- Tables are in namespaces
- Map/reduce with Hadoop

Best used: If you need a better HBase. 需要一个更好的HBase

For example: Same as HBase, since it's basically a replacement: Search engines. Analysing log data. Any place where scanning huge, two-dimensional join-less tables are a requirement.

Graph databases

图像数据库

OrientDB (2.0)

- **Written in:** Java
- **Main point:** Document-based graph 基于文档的图像

Neo4j (V1.5M02)

- **Written in:** Java
- **Main point:** Graph database - 图像数据库 - 连接的数据

database
- **License:** Apache 2.0
- **Protocol:** binary, HTTP REST/JSON, or Java API for embedding
- Has transactions, full ACID conformity
- Can be used both as a document and as a graph database (vertices with properties)
- Both nodes and relationships can have metadata
- Multi-master architecture
- Supports relationships between documents via persistent pointers (LINK, LINKSET, LINKMAP, LINKLIST field types)
- SQL-like query language (Note: no JOIN, but there are pointers)
- Web-based GUI (quite good-looking, self-contained)
- Inheritance between classes. Indexing of nodes and relationships
- User functions in SQL or JavaScript
- Sharding
- Advanced path-finding with multiple algorithms and Gremlin traversal language
- Advanced monitoring, online backups are commercially licensed

Best used: For graph-style, rich or complex, interconnected data.

For example: For searching routes in social relations, public transport links, road maps, or network topologies.

connected data
- **License:** GPL, some features AGPL/commercial
- **Protocol:** HTTP/REST (or embedding in Java)
- Standalone, or embeddable into Java applications
- Full ACID conformity (including durable data)
- Both nodes and relationships can have metadata
- Integrated pattern-matching-based query language ("Cypher")
- Also the "Gremlin" graph traversal language can be used
- Indexing of nodes and relationships
- Nice self-contained web admin
- Advanced path-finding with multiple algorithms
- Indexing of keys and relationships
- Optimized for reads
- Has transactions (in the Java API)
- Scriptable in Groovy
- Clustering, replication, caching, online backup, advanced monitoring and High Availability are commercially licensed

Best used: For graph-style, rich or complex, interconnected data.

For example: For searching routes in social relations, public transport links, road maps, or network topologies.

The "long tail" (Not widely known, but definitely worthy ones)

长尾理论

Couchbase (ex-Membase) (2.0)

- **Written in:** Erlang & C

Riak (V1.2)

- **Written in:** Erlang & C, some JavaScript
- **Main point:** Fault tolerance 容错功能

Memcached兼容，但具有持久化和集群功能

- **Main point:** Memcache compatible, but with persistence and clustering
- **License:** Apache
- **Protocol:** memcached + extensions
- Very fast (200k+/sec) access of data by key 超快的键数据访问速度(20w+每秒)
- Persistence to disk 持久化到磁盘
- All nodes are identical (master-master replication) 所有节点都是一样的(主-主复制)
- Provides memcached-style in-memory caching buckets, too 提供Memcached方式的内存缓存桶
- Write de-duplication to reduce IO 写入重复数据删除
- Friendly cluster-management web GUI
- Connection proxy for connection pooling and multiplexing (Moxi) 基于连接池和多路复用的连接代理
- Incremental map/reduce
- Cross-datacenter replication 跨数据中心复制

Best used: Any application where low-latency data access, high concurrency support and high availability is a requirement. 低时延、高并发支持和高可用

For example: Low-latency use-cases like ad targeting or highly-concurrent web apps like online gaming (e.g. Zynga). 公告定向投放系统 在线游戏

Scalaris (0.5)

- **Written in:** Erlang
- **Main point:** Distributed P2P key-value store 分布式的P2P键-值存储
- **License:** Apache
- **Protocol:** Proprietary & JSON-RPC
- In-memory (disk when using Tokyo Cabinet as a backend) 内存缓存
- Uses YAWS as a web server
- Has transactions (an adapted Paxos commit) 一致性、分布式的写操作
- Consistent, distributed write operations
- From CAP, values Consistency over Availability (in case of network partitioning, only the bigger partition works) 值的一致性胜过可用性

Best used: If you like Erlang and wanted to use Mnesia or DETS or ETS, but you need

- **License:** Apache
- **Protocol:** HTTP/REST or custom binary
- Stores blobs
- Tunable trade-offs for distribution and replication
- Pre- and post-commit hooks in JavaScript or Erlang, for validation and security.
- Map/reduce in JavaScript or Erlang
- Links & link walking: use it as a graph database
- Secondary indices: but only one at once
- Large object support (Luwak)
- Comes in "open source" and "enterprise" editions
- Full-text search, indexing, querying with Riak Search
- In the process of migrating the storing backend from "Bitcask" to Google's "LevelDB"
- Masterless multi-site replication and SNMP monitoring are commercially licensed

Best used: If you want something Dynamo-like data storage, but no way you're gonna deal with the bloat and complexity. If you need very good single-site scalability, availability and fault-tolerance, but you're ready to pay for multi-site replication.

For example: Point-of-sales data collection. Factory control systems. Places where even seconds of downtime hurt. Could be used as a well-update-able web server.

VoltDB (2.8.4.1)

- **Written in:** Java
- **Main point:** Fast transactions and rapidly changing data 超快事务和快速的变更数据
- **License:** GPL 3
- **Protocol:** Proprietary 内存的关系型数据库
- In-memory **relational** database.
- Can export data into Hadoop
- Supports ANSI SQL 支持标准的SQL

something that is accessible from more languages (and scales much better than ETS or DETS).

For example: In an Erlang-based system when you want to give access to the DB to Python, Ruby or Java programmers.

Aerospike (3.4.1)

- **Written in:** C
- **Main point:** Speed, SSD-optimized storage 速度：SSD优化存储
- **License:** License: AGPL (Client: Apache)
- **Protocol:** Proprietary
- Cross-datacenter replication is commercially licensed
- Very fast access of data by key 通过键快速访问数据
- Uses SSD devices as a block device to store data (RAM + persistence also available) 使用SSD作为存储数据的块设备(RAM+持久化)
- Automatic failover and automatic rebalancing of data when nodes added or removed from cluster 自动故障转移、集群节点数据自平衡
- User Defined Functions in LUA
- Cluster management with Web GUI Web集群管理
- Has complex data types (lists and maps) as well as simple (integer, string, blob) 丰富的数据类型(列表、映射表、整数、字符串)
- Secondary indices 二级索引
- Aggregation query model 聚合查询
- Data can be set to expire with a time-to-live (TTL)
- Large Data Types 低延时数据访问、高并发支持、高可用

Best used: Any application where low-latency data access, high concurrency support and high availability is a requirement.

For example: Storing massive amounts of profile data in online advertising or retail Web sites.

- Stored procedures in Java
- Cross-datacenter replication 跨数据中心的复制

Best used: Where you need to act fast on massive amounts of incoming data.

For example: Point-of-sales data analysis. Factory control systems. 销售点数据分析 工厂控制系统

Kyoto Tycoon (0.9.56)

- **Written in:** C++
- **Main point:** A lightweight network DBM
- **License:** GPL
- **Protocol:** HTTP (TSV-RPC or REST)
- Based on Kyoto Cabinet, Tokyo Cabinet's successor
- Multitudes of storage backends: Hash, Tree, Dir, etc (everything from Kyoto Cabinet) 100万以上每秒的插入/选择操作
- Kyoto Cabinet can do 1M+ insert/select operations per sec (but Tycoon does less because of overhead)
- Lua on the server side
- Language bindings for C, Java, Python, Ruby, Perl, Lua, etc
- Uses the "visitor" pattern 热备份、异步复制
- Hot backup, asynchronous replication
- background snapshot of in-memory databases 内存数据库的快照存储
- Auto expiration (can be used as a cache server) 自动过期(作为一个缓存系统)

Best used: When you want to choose the backend storage algorithm engine very precisely. When speed is of the essence. 需要非常精确地选择后端的存储算法引擎、速度至关重要

For example: Caching server. Stock prices. Analytics. Real-time data collection. Real-time communication. And wherever you used memcached before. 缓存服务器、股票价格、分析系统、实时数据收集、实时通讯系统、所有原先使用Memcached的地方

Of course, all these systems have much more features than what's listed here. I only wanted to list the key points that I base my decisions on. Also, development of all are very fast, so things are bound to change.