Datomic

Data Differently

What is Datomic?

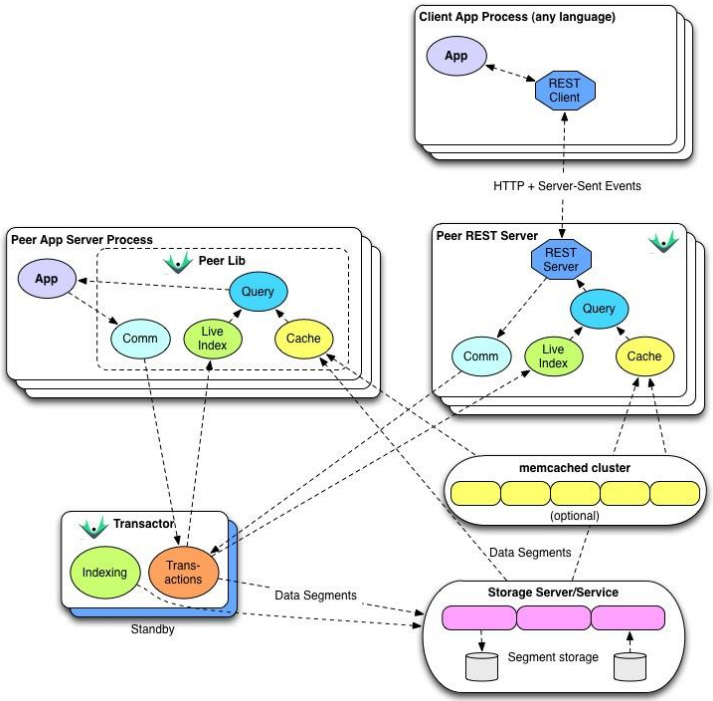
A new type of database providing:

- Immutable data
- Full ACID transaction support
- Rich schema and query capabilities on top of modern scalable storage engines
- Query-able History!!
- Designed to be easily distributed/scalable.

Architecture

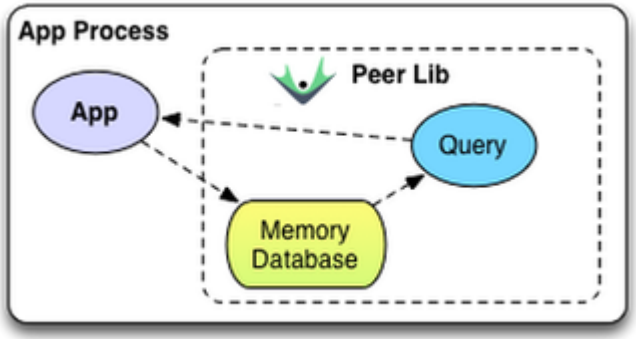
3 components

1. Peer Library
2. Transactor
3. Storage Service



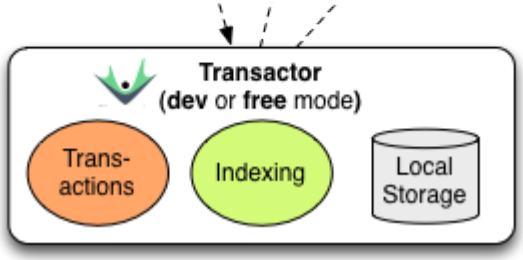
Peer Library

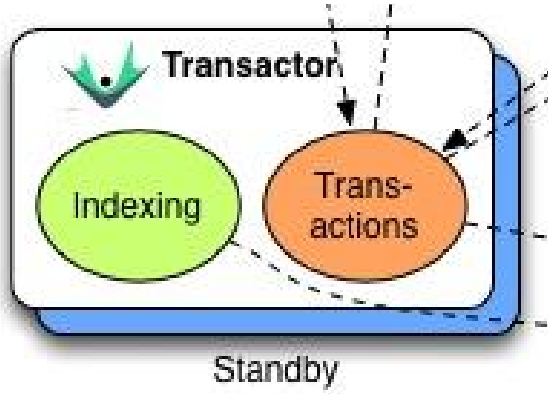
- Embedded in your application server(s).
- Gateway to the rest of the database, submitting transactions and accepting changes from the transactor.
- Provides data access, caching and query capability.
- Contains all the communication components needed for connecting to the transactor and storage services, as well as Datalog and other facilities for managing your data.
- Can act in standalone mode, using an in-memory database as a stand-in for the other components



The Transactor

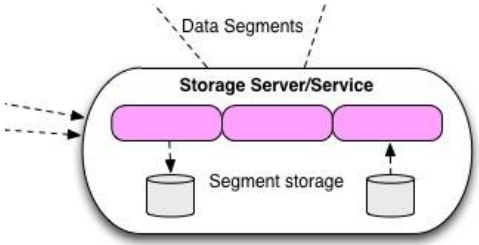
- A process that accepts transactions and coordinates change.
- Needed whenever you want a system comprised of more than a single peer, or one that leverages a storage service.
- Can be run as a stand-alone process, or you can run our prebuilt AMI on EC2
- With a transactor in play, the system can run in one of several storage modes.
- Commit the results to the storage service.
- All writes are synchronously made to redundant storage.
- Transmit changes to the peers.
- Index in the background, place indexes in storage service.





Storage Service

- Provide an interface to highly reliable, redundant storage.
- Available from third-parties.
- The currently supported protocols are **ddb** backed by DynamoDB, **riak** backed by Riak, **cass** backed by Cassandra, **couchbase** backed by Couchbase, **inf** backed by Infinispan, or **sql** backed by a SQL database.



Architecture Benefits

Many benefits are achieved with this architecture

Separating reads and writes

When reads are separated from writes, writes are never held up by queries. The transactor is dedicated to transactions, and need not service reads at all

Integrated data distribution

Each peer and transactor manages its own local cache of data segments, in memory. This cache self-tunes to the working set of that application. All caching is fully integrated into the system, not added on top of the system manually by the users.

Every peer gets its own brain (query engine and cache)

Moving a proper, index-supported, declarative query engine into applications will enable them to work with data at a higher level than ever before, and at application-memory speeds.

Architecture Benefits cont..

Elasticity

Putting query engines in peers makes query capability as elastic as the applications themselves. In addition, putting query engines into the applications themselves means they never wait on each other's queries.

Ready for the cloud

All components are designed to run on commodity servers, with expectations that they, and their attached storage, are ephemeral.

The speed of memory

While the (disk-backed) storage service constitutes the data of record, the rest of the system operates primarily in memory.

Data Model

Immutable Data

- Datomic is built upon the model of data consisting of immutable values.
- Incorporating time in data allows the past to be retained (or not), and supports point-in-time queries.
- Datomic is a database of facts.
- Immutable data means strong consistency combined with horizontal **read** scalability, plus built-in caching.
- **Writes** are not update-in-place; all data is retained by default. This provides built-in auditing and the ability to query history.

Atomic Data - the Datom

- Once you are storing facts, it becomes imperative to choose an appropriate granularity for facts.
- A datom consists of an entity, attribute, value and transaction (time).
- Any of those sets can be discovered via query, without embedding them into a structural storage model that must be known by applications.

Data Model cont..

Minimal Schema

- The schema required to encode datoms is extremely minimal, consisting primarily of the attribute definitions, which specify name, type, cardinality etc.
- Applications written to this model are free of the structural rigidity of relational and document models.

The Database

- A database is just a set of datoms, indexed in various ways.
- These indexes contain all of the data, not pointers to data.
- The storage service and caches are just a distribution network for the data segments of these indexes, all of which are immutable, and thus effectively and coherently cached.

Programming Model

Connecting

- A peer embedded in an application connects to a storage service and transactor.
- Will pull index/data segments from the storage service as needed, and cache them locally, and will get updates from the transactor.
- Queries and data operations work against a dynamically merged view of the world (the stable index + recent changes).
- After running for a while, the working set of the application will be cached locally, causing the majority of queries to incur no network activity at all.

Database programming with ... data!

- Traditional databases have data manipulation and query languages based upon strings.
- Datomic is designed to be a programmable database, and as such uses widely supported data structures like lists and maps for transactions and queries.
- These data structures are far easier to construct and compose programmatically than are strings. Query results are returned as similar data.

Programming Model cont..

Transactions

- Transactions fundamentally assert or retract sets of facts.
- A set of assertions/retractions/functions, represented as data structures, is sent to the transactor as a transaction, and either succeeds or fails as one.

Consistency

- Applications always work on a completely consistent snapshot of the database.
- The same snapshot can be used for an arbitrary time, for multiple queries etc, in full confidence that all the results will correlate, regardless of what has transpired in the interim.
- All this occurs without impeding other peers, or even other threads using the database in the same process.

Time

- Given a value of the database, one can obtain another value of the database as-of, or since, a point in time in the past, or both (creating a windowed view of activity).
- These database values can be queried ordinarily, without having to make special queries parameterized by time.

Queries

Datomic supports **Datalog** as a query language. Datalog is a deductive query system combining a database of facts (the Datomic db) with a set of rules for deriving new facts from existing facts and other rules.

Datalog is a great fit for application queries due to

- Pattern-matching like structure, joins are implicit
- Recursion is much more straightforward than in SQL
- Datalog rules subsume SQL views, but have more of a logic feel, allowing a closer alignment to business rules

Datomic's Datalog can query both database and non-database sources, and more than one source together, allowing transparent extension of declarative programming to application data. Because the query engine runs locally, these functions can be arbitrarily complex and run much more safely than having to install them on some server.

Demo

Time for some REPL action...

URLs

- http://www.datomic.com/
- https://github.com/Datomic/day-of-datomic
- http://docs.datomic.com/
- https://github.com/CraZySacX/glitter
-