Relational MySQL vs Cassandra NoSQL

Ken Smith

NoSQL choices

Keyvalue stores
Dynamo, Voldemort, Citrusleaf, Membase, Riak, Tokyo Cabinet, FoundationDB

“Bigtable” clones
Google Bigtable, Cassandra, HBASE, Hypertable

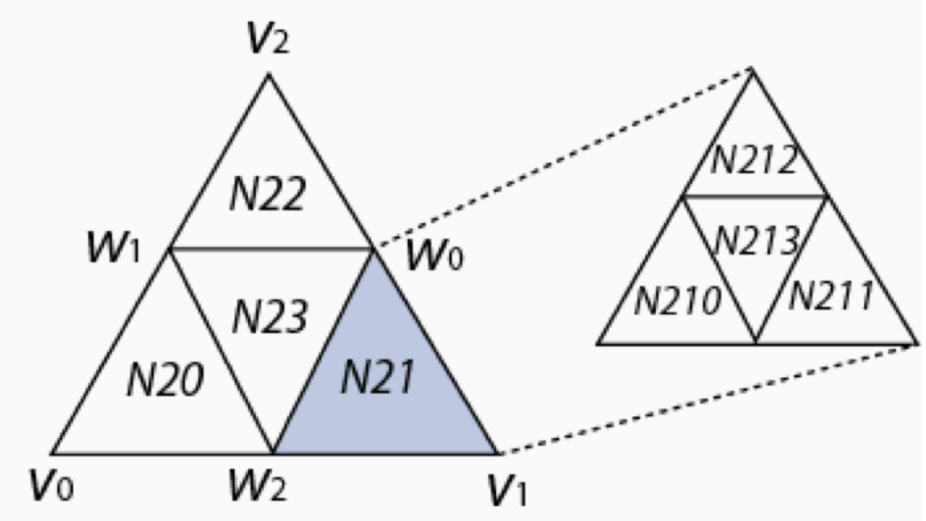
Document databases
CouchOne, MongoDB, Terrastore, OrientDB

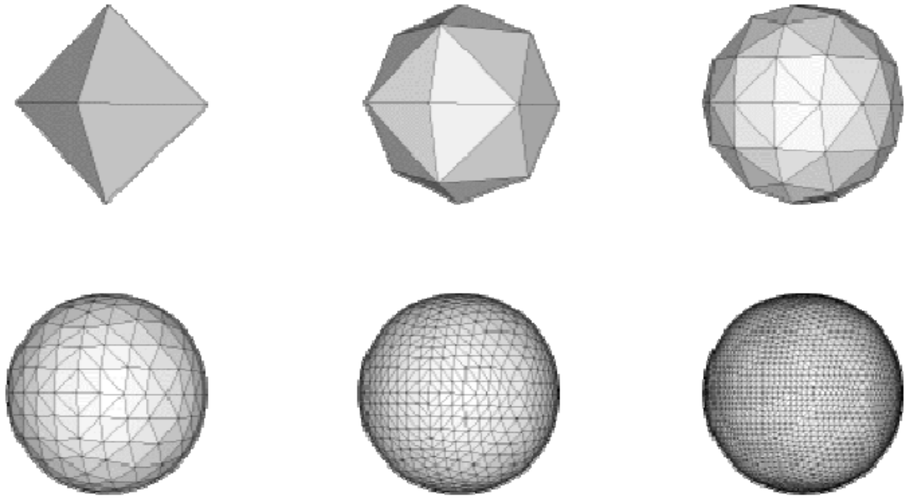
Graph databases
FlockDB, AllegroGraph, DEX, InfoGrid, Neo4J, Sones).

SQL choices

MySQL, MariaDB + HTM spatial indexing (not built in)
PostgreSQL (+ Q3C spatial indexing - custom built)
SQL Server (+ HTM)
Oracle

Cone Searching & HTM





Hierachical Triangular Mesh - quad tree spatial indexing system - celestial sphere divided into triangles.
Each triangle bounded by great circles subdivided into four more triangles. See Dave's Talk.
Triangles close to each other spatially are also close numerically
Mapped onto binary (groups of 2 bits), but usually represented in decimal

Relational Databases & HTM

The HTM API returns RANGES of triangles within specified radius (e.g. 5 arcsec)
Here's a small C++ program to return triangle ranges given an RA, Dec and a radius in arcsec.
Results are returned in decimal ranges.

(In this case, the values are 85.13154994520691, 37.936764657367384 and 5 arcsec)

```
./HTMCircle 16 85.13154994520691 37.936764657367384 5 tcs_cat_gaia_dr2
```

```
select * from tcs_cat_gaia_dr2 where
   htm16ID between 64759070916 and 64759070917
or htm16ID between 64759070919 and 64759070919
or htm16ID between 64759070936 and 64759070936
or htm16ID between 64759070938 and 64759070939
or htm16ID between 64759070960 and 64759070975
;
```

Relational Databases & HTM

Increase the radius to 50 arcsec:

```
./HTMCircle 16 85.13154994520691 37.936764657367384 50 tcs_cat_gaia_dr2

select * from tcs_cat_gaia_dr2 where
   htm16ID between 64758611968 and 64758612223
or htm16ID between 64758612352 and 64758612352
or htm16ID between 64758612864 and 64758612879
or htm16ID between 64758612896 and 64758612911
or htm16ID between 64758612916 and 64758612919
or htm16ID between 64759070720 and 64759071743
or htm16ID between 64759922688 and 64759922943
or htm16ID between 64759923264 and 64759923264
or htm16ID between 64759923520 and 64759923523
or htm16ID between 64759923526 and 64759923526
or htm16ID between 64759923534 and 64759923534
;
```

Cassandra & HTM

Cassandra has a query language called CQL, which is SQL like, with the following exceptions

OR statements are not allowed - but IN statements ARE allowed. Ranges are NOT allowed.

Let's do the 5 arcsec query again, but this time we expand out all the triangles.

```
./HTMCircleAllIDs 16 85.13154994520691 37.936764657367384 5 tcs_cat_gaia_dr2
```

```
select * from tcs_cat_gaia_dr2 where htm16ID IN (
64759070916,64759070917,64759070919,64759070936,64759070938,
64759070939,64759070960,64759070961,64759070962,64759070963,
64759070964,64759070965,64759070966,64759070967,64759070968,
64759070969,64759070970,64759070971,64759070972,64759070973,
64759070974,64759070975);
```

Cassandra & HTM

Same query again, but increase the radius to 50 arcsec:

```
./HTMCircleAllIDs 16 85.13154994520691 37.936764657367384 50 tcs_cat_gaia_dr2

select * from tcs_cat_gaia_dr2 where htm16ID IN (
64758611968,64758611969,64758611970,64758611971,64758611972,
.
.
<1570 more triangles!!>
.
.
64759923521,64759923522,64759923523,64759923526,64759923534);
```

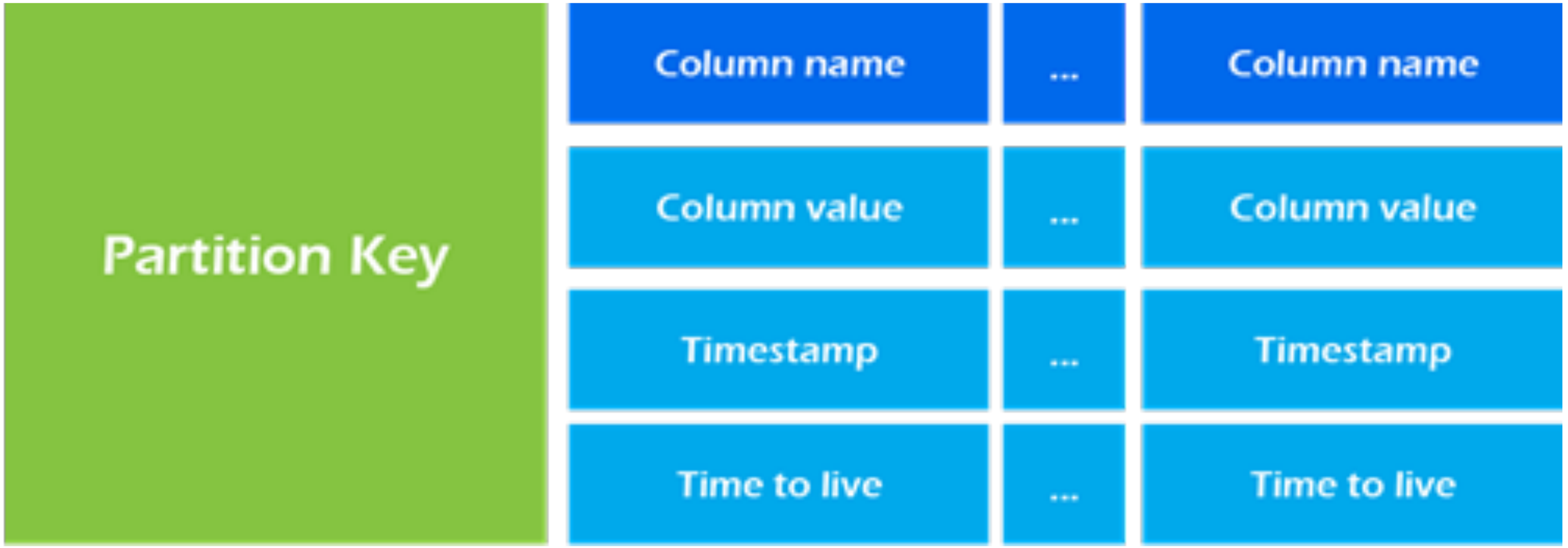
The query works, but it's getting very verbose, and doubling the radius again effectively quadruples the triangle count.

Cassandra

Cassandra Data Model	Relational Data Model
Keyspace	Database
Column Family	Table
Partition Key	Primary Key
Column Name/Key	Column Name
Column Value	Column Value

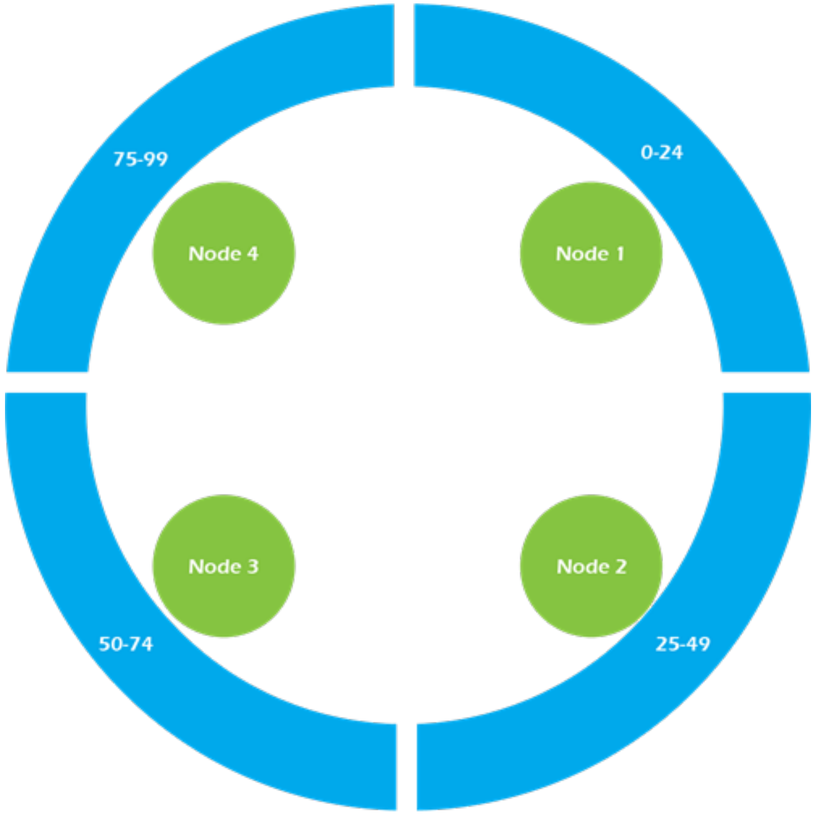
Cassandra is a Key - Value store
Flat tables - no relations
“Tables” are groups of columns or “Column Families”

Cassandra Partition Keys



Primary Keys can be more than one column.
First column is a Partition Key (tells which node to store the data)
Subsequent columns are Clustering Keys

Cassandra Replication



Replication follows a ring architecture
Partition Key determines which Node to which the data is primarily copied.
Replication factor determines how many nodes to which the data is copied.
Replication goes clockwise in this architecture
My installation - on laptop - just a single Node

Partition Keys & Clustering Keys

Cassandra will NOT allow you to query any old column with any constraint.
Build the column families (tables) with the query in mind

E.g. Gaia DR2 table. The unique identifier column is “source_id”.
This doesn’t mean much in terms of organising the data.

HTM16	source_id	ra	dec	phot_g_mean_mag
54680902005	4309278608071975424	288.70392	9.99498	17.5

Primary Key is now HTM16 AND source_id
Cone search by pulling out a list of HTM16s associated with an RA and Dec
But - as above HTM level 16 IDs explode above a few 10s of arcsec

Additionally - we can’t add other HTM levels (like in SQL), because we can only search on the primary column.

Putting the H in HTM

HTM is Hierarchical
Level 16 is a superset of Level 13
Level 13 is a superset of Level 10

	Decimal	Binary	Base4
HTM16	54680902005	**1100101110110011110001011**00101110101	**N023230330112**11311
HTM13	854389093	**1100101110110011110001011**00101	**N023230330112**11
HTM10	13349829	**1100101110110011110001011**	**N0232303301**1

Use the string representation (base 4) representation
Split the deeper HTM levels into suffixes

Primary Key + Clustering Keys

HTM10	HTM13	HTM16	source_id	ra	dec	phot_g_mean_mag
N02323033011	211	311	4309278608071975424	288.70392	9.99498	17.5

Query using the HTM10 field on its own (the partition key),
the HTM10 and HTM13 fields (= HTM level 13),
the HTM10 and HTM13 and HTM16 fields (= HTM level 16).
(Full set of primary key fields includes the source_id.)
Can't query via a clustering key out of order

New queries based on clustering keys

HTM10 query

```
select * from tcs_cat_gaia_dr2 where htm10 IN ('S23023222101','S23023222132');
```

HTM13 query

```
select * from tcs_cat_gaia_dr2 where htm10 IN ('S23023222101','S23023222132')
AND htm13 IN
('000','001','002','003');
```

HTM16 query

```
select * from tcs_cat_gaia_dr2 where htm10 IN ('S23023222101','S23023222132')
AND (htm13,htm16) IN
(('000','100'),('001','200'),
('003','200'),('000','200'),
('002','100'),('003','100'));
```

Loading up Gaia DR2 data into Cassandra & 1 million queries

As with HDD vs SSD:

55 million rows of Gaia DR2 data loaded into Cassandra using this partition & clustering scheme.
Took 25 hours! (About 600 inserts per second.)
But inserts were being done single threaded, and not in “batch” mode.

Cone search for 1,000,000 random rows based on the HTM level 16 (0.5 arcsec radius)
1 hour and 50 minutes - or 6,600 seconds.
3 times slower than the HDD timings
BUT not yet using Cassandra to its full capacity, & searching done single-threaded.

Numbers SHOULD massively improve with properly distributed Cassandra system.
Various statements online indicate that (e.g.) a 15 node cluster can cope with 120,000 inserts / sec.

Conclusions & Further work

Cassandra CAN be used for data loading and cone searching using the mechanisms above.

Additionally the HTM interface also allows for other types of spatial searches - not just cones. All we need are the triangle IDs.

Primary keys need to be chosen very carefully!

Extraction of data based on ranges of other columns require that the data be copied into separate tables and primary keys chosen accordingly.

This should be deployed as a group of machines into a properly distributed infrastructure (e.g. a group of openstack machines) to test both loading speed AND query speed.

Other relational technologies to be considered

CitusData - bought by Microsoft

PostgreSQL with a distributed architecture, not unlike Qserv.

PostgreSQL used extensively in Gaia - with Q3C spatial indexing.