instaclustr

VNodes… and 4.0

Agenda

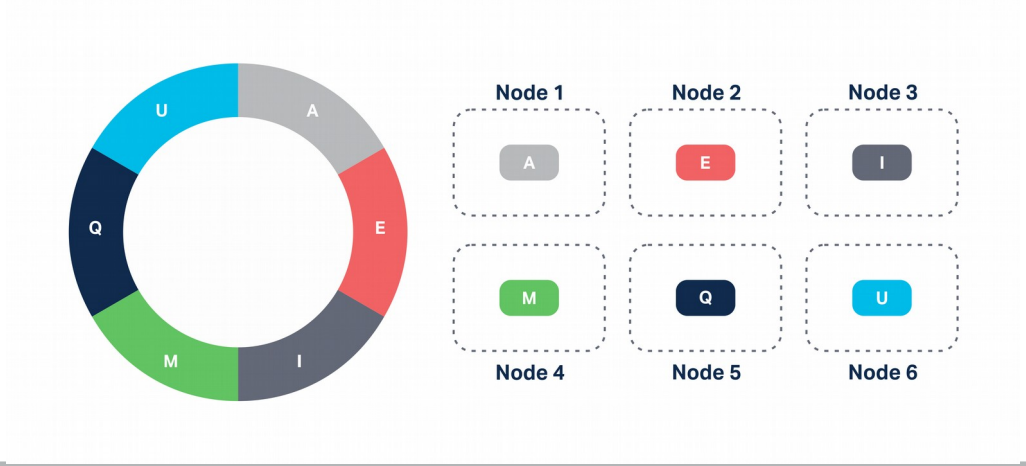
A HISTORY OF VNODES

- The Single Token Life
- What are vnodes (very briefly) and why we implemented them
- When, and counter solutions
- What can JIRA tell us?
- Some issues since inception
- 4.0 and the future of VNodes

Before VNodes

- Single token range per node
  - Manual token selection
  - Complex to scale out
  - Limited distribution factor

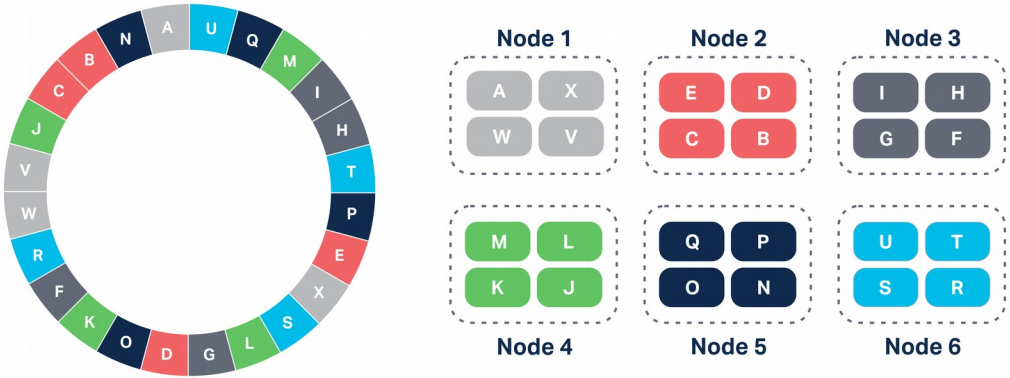
$$DF = 2RF - 2$$



instaclustr

What are VNodes?

- Lets a node own multiple token ranges
- Yaml defaults to 256 tokens from 2.0 onwards, but 1 if unspecified
- Spreads a single nodes replicas across the ring



instaclustr

WHY?

- Simplify adding and removing nodes
- Faster bootstrap and rebuild
- Better fault tolerance for streaming
- Better load distribution for streaming
- Better support for heterogeneous nodes

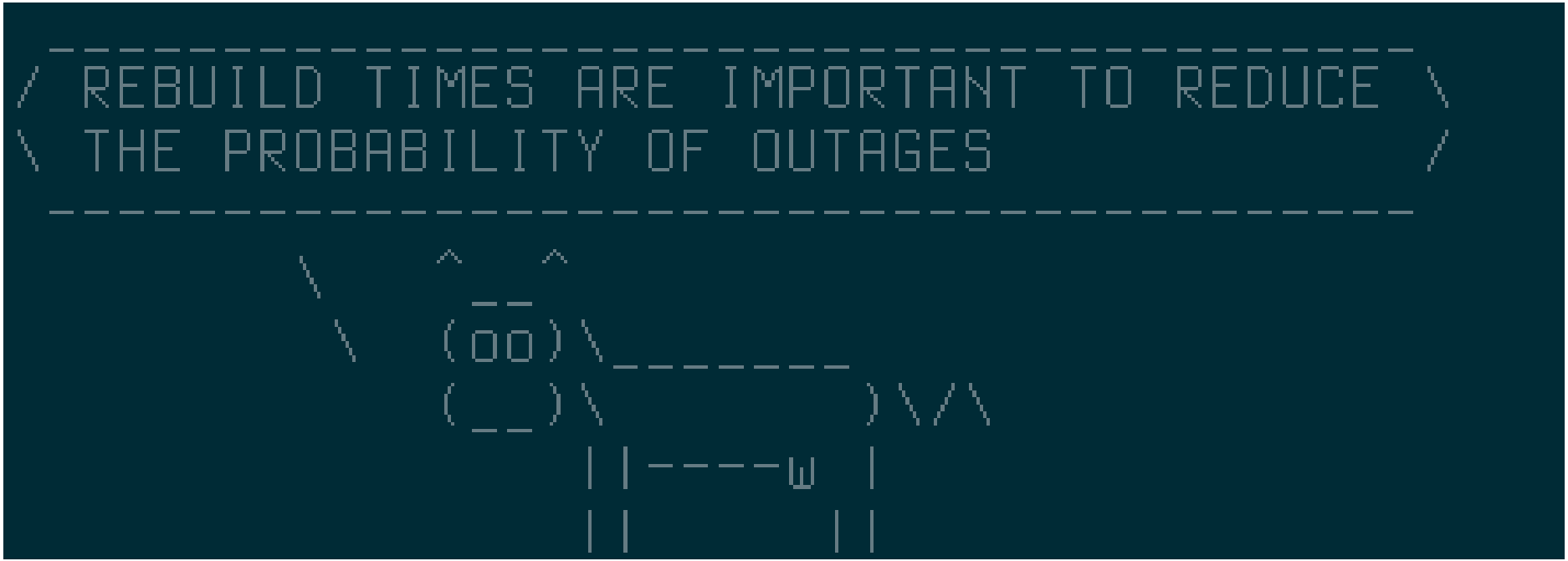
Remember, this was all back in the days before the “cloud”!

Old news?

```
It seems that since (and correct me if I'm wrong here) every physical node
will likely share some small amount of data with every other node, that as
the count of physical nodes in a Cassandra cluster increases (let's say
into the triple digits) that the probability of at least one failure to
Quorum read/write occurring in a given time period would *increase*.
```

Subject: Virtual Nodes, lots of physical nodes and potentially increasing outage count?

Doesn’t matter, Cow’s opinion



1.2

- Big problem in the community. Raised in many tickets and ML threads before implementation
- Counter-solutions:
  - CRUSH/Distribution Factor based solution. Decouple Distribution Factor from RF and N
  - Available token range + desired token range, with a gradual move to the desired range.
- DF = N was perceived optimal
- Tuneable DF partially achieved through NTS and racks + DC's

$$Single\ Token\ DF = 2\ RF - 2$$

$$VNodes\ DF = N$$

The Change

32 files changed, 2719 insertions(+), 343 deletions(-)

- Decent, but not MASSIVE change
- ~1600 insertions and 14 deletions were the Shuffle utility
- StorageService.java in comparison was 3649 LOC, codebase was ~143000 LOC
- Primarily developed by a few folk from Acunu

A quick analysis of JIRA

Total vnode related bugs that have been resolved	59
Total w/ unresolved	74
Total bugs that *mention* vnodes	98
Total issues that *mention* vnodes	198
Total issues that *mention* range tombstones	158

Version	1.2	2.0	2.1	2.2	3.0	3.x
# of related, resolved bugs	16	18	22	12	23	28
# of mentions	29	46	55	23	46	66

Note: excludes <=1.2.0

Someone saw it coming...

1) vnodes add enormous complexity to *many* parts of Cassandra. I'm skeptical of the cost:benefit ratio here.

1a) The benefit is lower in my mind because many of the problems solved by vnodes can be solved "well enough" for "most people," for some value of those two phrases, without vnodes.

Shuffle

- Migration to VNodes with Shuffle
- Increase num_tokens and shuffle for a balanced cluster
- Too good to be true?

Shuffle (LANGUAGE WARNING)

If your shuffle succeeds, you will be the first reported case of shuffle succeeding on a non-test cluster. Until I hear a report of someone having real world success, I recommend against using shuffle.

I really *DON'T* want to deal with another shuffle....but what options do I have, since vnodes "make it unneeded to balance the cluster"? (which, at the moment, seems like a load of bullshit).

AFAIK shuffle was not an easy thing to test and does not get much real world use as only some people will run it and they (normally) use it once.

I am still afraid of this step. Yet you can avoid it by introducing new nodes, with vnodes enabled, and then remove old ones. This should work.

My problem is that I am not really confident in vnodes either...

I had a situation earlier where my shuffle failed after a hard disk drive filled up. I went through and disabled shuffle on the machines while trying to get the situation resolved. Now, while I can re-enable shuffle on the machines, when trying to do an ls, I get a timeout.

Sounds like your cluster got shufflef*cked.

We recently completed a migration from a production cluster not using vnodes and in a single DC to one using vnodes in two DCs. We used the "just spin up a new DC and rebuild" strategy instead of shuffle and it worked. The checklist was long but it really wasn't more complicated

We had to bail on shuffle since we need to add capacity ASAP and not in 20 days.

After 2 days of endless compactions and streaming I had to stop this and cancel shuffle. One of the nodes even complained that there's no free disk space (grew from 30GB to 400GB). After all these problems number of the moved tokens were less than 40 (out of 1280!).

It will perhaps be some consolation to hear that this insane misfeature of automatic token assignment by range bisection is finally going away in Cassandra 2.0.

Just a followup on this issue. Due to the cost of shuffle, we decided not to do it. Recently, we added new node and ended up in not well balanced cluster:

We recently upgraded from version 1.1 to 1.2
It all went well, including setting up vnodes, but shuffle fails.

adding nodes and then removing nodes appears to be a faster way to shuffle data for small clusters. Obviously not always possible, but I thought I'd just throw this out there in case anyone runs into a similar situation.

Repairs

- Single Tokens? Easy!
  - Pick non-overlapping nodes and repair ‘til your hearts content
- Many performance problems – Iterating over all those vnodes sure does hurt :(
  - Mostly addressed in many performance improvements patches, 2.2, and 3.0
- VNodes? All that load balancing and fault tolerance isn’t helping you now!
- SSTABLE EXPLOSIONS!
- Incremental repair really didn’t help. Anti-compaction hell and CASSANDRA-9143
- Impossible except for subrange repairs coordinated by some external system

Load balancing

- java.util.Random not as much entropy as we'd hoped :(
- Somewhat resolved in 3.0 with CASSANDRA-7032 but still tedious

```
Load          Tokens       Owns (effective)
266.08 GB     256          45.2%
258.24 GB     256          49.7%
271.32 GB     256          49.8%
274.84 GB     256          50.2%
278.71 GB     256          50.3%
272.08 GB     256          54.8%
```

Vnodes and 4.0

- Streaming improvements (CASSANDRA-12229, CASSANDRA-4650)
  - Non-blocking IO + Netty give 15-20% streaming performance improvement
  - Stream from more replicas when N>=3 per DC – helps bootstrap and replace times
- Repair merkle tree comparison (CASSANDRA-3200)
  - Less overstreaming
- Incremental repair re-write (CASSANDRA-9143)
  - Moves anti-compactions to start of repair
  - May make incremental repair viable
  - Still doesn’t solve the anti-compaction problem :(

Future<Cassandra>

Subject: Cassandra Needs to Grow Up by Version Five!

Finish the basic coding of Cassandra, make it easy to use for administrators, make is smart, add cluster wide management. Keep Cassandra competitive or it will soon be the old Model T we all remember fondly.

- Kenneth Brotman

Future<Vnodes>

- RangeAwareCompaction
  - Say no to anti-compactions
  - Zero copy streaming of SSTables :drool:
  - Bonus: Deleting/updating old data is now E Z
  - Bonus: Low SSTables per Read
- JBOD (maybe?)
  - Fix the system table problem….
- Fix the defaults (basic coding)

Questions and Doubtful points?