Mar
19
Riak and Cassandra

Please don't read this without also reading about the social context. I stand by the technical content, but it was worded more harshly than it should have been and in particular the attribution of motive was wrong. I've already admitted as much to the Basho folks in person, and I've already apologized. Let's move on.

One of the most annoying things about many NoSQL projects is the relentless promotion of certain projects. The competition for users, contributors, potential investors and customers, speaking and consulting engagements, and general attention is fierce. People really really want projects that they're involved with to succeed, and I can respect that they're willing to fight for that . . . but sometimes they fight dirty and I don't respect that so much. Sadly, one example that recently appeared is Basho's comparison of their own product Riak to Cassandra. Now, I know the Basho guys aren't stupid. Riak does basically work, and stupid people wouldn't have gotten it that far. Some of the explanations on their site of things like consistent hashing and vector clocks are quite good. Even the article I'm about to address demonstrates that they actually do know a lot about this stuff . . . so ignorance is not a likely excuse for its misrepresentations. They must have known their comparisons were inaccurate, I know some of the inaccuracies have been addressed by others, they've had ample time to make corrections, and yet the misrepresentations remain. Let me address a few so people can see what I mean.

> When you add a new node [in Riak], it immediately begins taking an equal share of the existing data from the other machines in the cluster, as well as an equal share of all new requests and data. This works because the available data space for your cluster is divided into partitions (64 by default).
> …
> When you add a machine to a Cassandra cluster, by default Cassandra will take on half the key range of whichever node has the largest amount of data stored. Alternatively, you can override this by specifying an InitialToken setting, providing more control over which range is claimed. In this case data is moved from the two nodes on the ring adjacent to the new node. As a result, if all nodes in an N-node cluster were overloaded, you would need to add N/2 new nodes. Cassandra also provides a tool ('nodetool loadbalance') that can be run in a rolling manner on each node to rebalance the cluster.

Quick question: does dividing keys into 64 partitions give as much load-balancing flexibility as dividing at any point in the 128-bit space provided by MD5 (which is what you'd have with Cassandra's RandomPartitioner)? 64 is a particularly problematic number, not only because of what happens if you have more than 64 nodes but because even with fewer nodes the granularity is just too coarse. If a partitition is overloaded, too bad. There's even a comment in the admin documentation that comes with Riak saying that you should set this to several times the number of nodes in your cluster . . . as though that's static. What if you set it appropriately for a four-node cluster and then grew to forty? You won't find them discussing those issues in the comparison but, oh, they sure are quick to speculate about needing to add N/2 nodes with Cassandra. At least they mention "nodeprobe loadbalance" but they get the command wrong and don't seem to appreciate what it really does. Go read CASSANDRA-192 for yourself, and you'll see that it can actually balance load much better than Riak with its partitions ever could.

> Cassandra has achieved considerably faster write performance than Riak.
> ….
> That said, the Riak engineering team has spent a lot of time optimizing Riak performance and

benchmarking the system to ensure that it stays fast under heavy load, over long periods of time, even in the 99th-percentile case. We like speed, just not at the expense of reliability and We like speed, just not at the expense of reliability and scalability.

In other words, "We're forced to admit that Cassandra is faster but we'll spread FUD to distract from that." I'm pretty sure the Cassandra folks are also unwilling to compromise on reliability and scalability for the sake of speed. The authors of the comparison have in no way shown that Riak even has an advantage in reliability or scalability, but their conclusion is worded to imply – without actually having the guts to state outright – that they made a responsible tradeoff and Cassandra made an irresponsible one. That's just disgusting.

> Riak tags each object with a vector clock, which can be used to detect when two processes try to update the same data at the same time, or to ensure that the correct data is stored after a network split.
> …
> In contrast, Cassandra tags data with a timestamp, and compares timestamps to determine which data is newer. If a client's timestamp is out of sync with the rest of the cluster, or if a client waits too long between reading and writing data, then it is possible to lose the data that was written in between.

The difference between timestamps and vector clocks is a legitimate one, and in this case I think the Riak folks are right to bring it up. I personally would prefer a vector-clock-based approach. The "waits too long . . . in between" is kind of FUD-ish, though. This is a potential problem in any eventually consistent system, including those that use vector clocks (which resolve some but not all conflicts). Riak does provide the building blocks for a good solution, in the form of their X-Riak-Vclock extension and user-driven conflict resolution, but they don't even allude to that. I guess "Cassandra might screw up your data" was easier than discussing a point that might actually have favored them.

> Riak buckets are created on the fly when they are first accessed. This allows an application to evolve its data model easily.
> …
> In contrast, the Cassandra Keyspaces and Column Families (akin to Databases and Tables) are defined in an XML file. Changing the data-model at this level requires a rolling reboot of the entire cluster.

Another completely fair point. Supercolumns have their uses, but they're really no substitute for dynamic bucket/ColumnFamily creation.

> In contrast, Cassandra has only one setting for the number of replicas

Absolutely, positively untrue. Cassandra in fact allows you to specify the number or replicas on a per request basis, to trade off performance vs. protection from failures. This is such a commonly discussed and central feature that I find it impossible to believe the authors weren't aware of it. Interpreting the abundant information on this topic as "only one setting" is, again, reprehensible.

As I hope readers can see, I'm not just rejecting every criticism of Cassandra. I like Cassandra, I like the developers, but I'm no fanboi even on projects I'm more directly involved in. There's always room for improvement. What I do object to, though, is criticism given without due diligence. I might even be wrong about Basho's load distribution, for example, but at least I tried to find the facts. I expect at least that much diligence and objectivity from people who are posting their findings on a company-sponsored website as part of their day jobs, and frankly I don't think those traits are very evident in

Basho's comparison.

This entry was posted on Friday, March 19th, 2010 at 9:49 pm and is filed under distributed. You can follow any responses to this entry through the RSS 2.0 feed. You can skip to the end and leave a response. Pinging is currently not allowed.
Comments

1. Jonathan Ellis Says: March 19th, 2010 at 10:51 pm

FWIW, online ColumnFamily creation + removal has been high on Cassandra's "stuff to fix" list for a while now, and Gary Dusbabek is implementing it for 0.7. You can see the progress in the finished "depends on" and "sub-tasks" in https://issues.apache.org/jira/browse/CASSANDRA-44 (I'd say he's about 80% done with the hard parts).

2. Jeff Darcy Says: March 20th, 2010 at 7:01 am

A question from Twitter:

> hornbeck: @spyced @obdurodon if we stated something wrong, why not just contact us? Why write a hateful post that helps no one? #riak #cassandra

Partly because I'm a mean, nasty person. Mostly because I have good reason to believe taking the polite route won't work. I have information that it was tried, by people more expert in these matters than me, and clearly didn't have an effect. That's hardly a surprise, since the nature of the misstatements is such that the claim of innocence strains credibility. Why didn't the authors of the comparison contact the Cassandra folks before they wrote their piece, like I'm apparently supposed to have done with them before I wrote mine? I'm sure they could have gotten answers. Perhaps they could even have turned it into a collaborative effort, which would have been good for everyone. Anybody who didn't even make that much effort to find the facts shouldn't even be publishing a comparison, and my response was no more "hateful" than the original, so spare me the aggrieved tone.

Some people reciprocate when others play by gentlemen's rules. Others take advantage until they see an elbow or two coming back at them. I've seen that in plenty of communities, both real and virtual. I've even been through this particular scenario before, quite recently, with Maxiscale. They took the opportunity to engage on the issues, they acquitted themselves well, I'm now fairly impressed with the directions they've taken, and I'm willing to dismiss the white paper in question as an aberration from an overzealous marketing type. (I will note, though, that it has been two months with no correction or retraction. Come on, guys!) If someone from Riak wants to take the high road and correct some of the inaccuracies in the comparison without being combative about it, good for them. I'll retract my speculation about their motives or character, and give them all due credit, if that happens. Let's see.

3. Jonathan Ellis Says: March 20th, 2010 at 8:39 am

The main reason Cassandra uses the load balancing scheme it does, rather than the "grab an even number of partitions from each existing node" that Riak does, is that the latter inherently means you get non-contiguous data on the new node(s). It does have the advantage of requiring less i/o initially, especially for small numbers of nodes, but without contiguous data ranges on each node you can't do

range queries performantly, and we think that's more important.

There's room for legitimate disagreement here; engineering is about tradeoffs, and there's no one-size-fits-all. But we were definitely aware of the riak-style approach (described in detail in the Dynamo paper) and deliberately went with something we think is better.

4.

Rusty Klophaus Says: March 20th, 2010 at 9:15 am

Hi Jeff, thanks for taking the time to read our comparison and write about it.

To address some of your comments:

The reason for discussing the "nodetool loadbalance" command was to point out that loadbalancing is automatic in Riak, while a manual, one-node-at-a-time operation in Cassandra. This is true as of February 9th, 2010 (http://www.mail-archive.com/cassandra-dev@incubator.apache.org/msg01489.html) but if this is incorrect, I'll gladly change the article in question. (Not sure what you mean when you say we got the command wrong. Cassandra's docs also refer to this as "nodetool loadbalance", http://wiki.apache.org/cassandra/Operations)

Regarding Performance and Vector Clocks, speed is a tradeoff, and explaining the details of that tradeoff, and the situations where that tradeoff becomes important, is not FUD.

Regarding Replicas, I am talking about the actual copies of the data stored on disk after a write operation completes. Riak can be configured, at a bucket level, to store one copy of some data, three copies of other data, and ten copies of a third kind of data, if you wish. In Cassandra, there is one setting (ReplicationFactor) that controls this for the entire cluster.

This is different from consistency settings/quorum paramaters, which are tunable on an operation level for both Riak and Cassandra. (ConsistencyLevel in Cassandra, or R-values, W-values, and DW-values in Riak).

That said, we're very open to correcting inaccuracies, and have already done so for each of the projects we posted about, based on feedback from well-known members in the community. We will continue to do so as we recieve more feedback, or as features evolve. The goal is to provide more clarity around the NoSQL space so that users can choose the best tool for their needs, whatever that may be.

Best,
Rusty

5.

Jeff Darcy Says: March 20th, 2010 at 1:41 pm

> loadbalancing is automatic in Riak, while a manual, one-node-at-a-time operation in Cassandra.

A completely fair point. I think most people would agree that automatic is generally preferable, though it's not a complete no-brainer since there are performance implications and some might want manual control over when to take the hit. Most people would also agree that the granularity of load balancing matters, as does automatic selection of the boundaries. The fact is that your approach has some advantages and Cassandra's has some advantages. The comparison makes it sound like Cassandra

is prone to these horrible load imbalances which limit scalability while Riak just cruises right along, when in fact several very-high-scale Cassandra users have shown that these issues are not a significant impediment.

Regarding Performance and Vector Clocks, speed is a tradeoff, and explaining the details of that tradeoff, and the situations where that tradeoff becomes important, is not FUD.

It wouldn't be, but trying to portray Cassandra as having pursued speed at the expense of reliability (with vector clocks as the solution) is. Cassandra has a different consistency model than Riak. It would even be reasonable to say it has a weaker one, but that's an issue of usability, not reliability. Even if they switched to vector clocks (CASSANDRA-580) and added custom or user-driven conflict resolution (which I've also seen discussed), there's ample reason to believe it would still be significantly faster than Riak. Surely that merits some reaction other than not-so-veiled accusations that the performance came at the expense of something else.

Regarding Replicas, I am talking about the actual copies of the data stored on disk after a write operation completes.

…

This is different from consistency settings/quorum paramaters, which are tunable on an operation level for both Riak and Cassandra.

One would be hard pressed to recognize that important distinction from your comparison. Yes, you gain a point on tunability there, but presenting Cassandra's functionality as "one setting for the number of replicas" without putting "number of replicas" in context is still wrong.

we're very open to correcting inaccuracies, and have already done so for each of the projects we posted about

I'll have to check the history to see if there were indeed changes I didn't notice, but of the things I knew to look for from my first reading none had changed when I wrote this last night.

6.

Jonathan Ellis Says: March 20th, 2010 at 3:02 pm

"In Cassandra, there is one setting (ReplicationFactor) that controls this for the entire cluster."

This is per Keyspace in 0.6 [beta3 just released].

7.

Mark Callaghan Says: March 20th, 2010 at 10:50 pm

They correctly claim that ReplicationFactor is fixed. You respond by describing support for per-write ConsistencyLevel. I think you should clarify this for the sake of your readers. I would much rather have what Cassandra provides — support for per-write ConsistencyLevel than for per-write or per-row ReplicationFactor.

8.

Rusty Klophaus Says: March 22nd, 2010 at 7:56 am

@Mark – to clarify, both Riak and Cassandra allow you to specify consistency levels on a per operation basis.

The distinction is on being able to set the number of copies of data stored on disk once the operation is complete. This is important because it directly affects how resilient your cluster is, ie: how many machine failures can your cluster withstand before data is unavailable.

In Riak, you can set the number of replicas on a bucket level. (Or, to describe things in Cassandra terms, you can specify a different ReplicationFactor per Keyspace.) As of Cassandra 0.5, there is only one ReplicationFactor setting for the entire cluster. As Jonathan pointed out, Cassandra 0.6 provides a ReplicationFactor setting per Keyspace.

@Jonathan – Thanks for the heads up, updated the Riak wiki to include this information.

9.

Jeff Darcy Says: March 22nd, 2010 at 8:15 am

Thanks for the clarification, Rusty. I kept meaning to get to that myself, but kept getting distracted.

As a matter of pure personal opinion, I don't think being able to set the replication factor separately for different data aggregations is all that useful. Why would people want more copies of data? Either for greater fault tolerance or to spread read load across the copies. For fault tolerance, very few people will go beyond two or three and those who do will want those copies spread across data centers. (Hmmm, maybe the comparison should include a section on rack or data center awareness in determining replica placement.) The read-load spreading issue only really applies when there's relatively high contention for a single object. I can sort of see that happening for a document-oriented store with large documents e.g. representing whole web pages, but it seems extremely unlikely for the sorts of data that Cassandra is built to handle. I just really can't see that much demand for higher replication factors in those environments. Do you even see much use of that feature among Riak users? Also, it's worth keeping in mind that one can simply run multiple separate servers (even on the same machine) with different replication factors. Since operations can't really span keyspaces anyway, no functionality is lost; there would be some small losses in administrative complexity and system-resource efficiency, but not so much that I think anyone need rule it out as an alternative.

As Mark points out, what really tends to matter more to most people is the fine grained control over R and W (in the Dynamo terminology) which is present in both Riak and Cassandra. That affects not the number of replicas in the system "at rest" but the amount of data "in flight" when a fault occurs – including data that hasn't been fully committed because of a partition. For most people, as long as data is sure to have hit disk somewhere, and eventually be replicated two or three places eventually, those are sufficient data-protection guarantees. After all, that's as good as databases and filesystems using asynchronous replication, and thousands upon thousands of paranoid IT folks seem satisfied with that. Data safety does become a serious issue in comparisons of Riak vs. memory-based stores, and even some disk-based ones such as MongoDB, but I really don't think it differentiates Riak vs. Cassandra.

10.

Jeff Darcy Says: March 22nd, 2010 at 4:36 pm

It looks like I managed to get Basho's attention. CTO Justin Sheehy posted The Craft Brewers of NoSQL (immediately praised by no fewer than four other Basho employees) talking about dialog among NoSQL folks and taking some not-so-veiled shots at yours truly in the process. As I pointed out in my Twitter response, though, positioning "makers and doers" vs. "trolls" might not qualify as taking the high road on matters of civility. I'll note here that VP of Engineering Andy Gross has already improved the comparison by removing some of the more inflammatory claims about how Cassandra achieved their performance (revisions 48 and 49 yesterday and today), while engineer Rusty Klophaus

has done so by incorporating information provided in this thread by Jonathan Ellis (revision 51 today). Kudos to you guys. This just goes to show that sometimes even “hateful” criticism can lead to constructive change. I don’t mind playing Bad Cop as long as the end result keeps getting better. That is what it’s all about, right?