Sharding Pinterest: How we scaled our MySQL fleet (pinterest.com)

144 points by jparise 154 days ago | past | web | 85 comments

devmach 154 days ago

As I read on their blog, Instagram uses similar logic with PostgreSQL. If anyone interested, it's accessible on http://instagram-engineering.tumblr.com/post/10853187575/sha...

abalone 154 days ago

That is an excellent compare & contrast. The both use 64 bit IDs.

Instagram: 13 bit shard ID, 51 bit "local" ID consisting of 41 bit timestamp in milliseconds and 10 bit sub-millisecond ID. So this scheme supports 1024 IDs per millisecond per shard for 41 years, and 8192 shards.

Pinterest: 16 bit shard ID, 10 bit type ID(?), 36 bit local ID, 2 bits reserved. This supports 68 billion objects per shard and 65K shards, but does not represent time. So you need another field / more storage for that. Also notable is the large 10 bit type ID field which seems to be only actually used for a handful of values, leading to a large chunk of bits that don't change across IDs.

In short, Instagram's scheme is more efficient largely due to the leverage of timestamps in the ID instead of type information.

boomzilla 154 days ago

It seems mysql (and hopefully postgresql sometime soon) with custom sharding logic in the app layer still hits the sweet spot for scaling to the order of 100M users. With some thoughts going into designing an appropriate data model and sharding logic, certain join queries can be delegated to the databases too.

sarnowski 154 days ago

Hi, at Zalando, we are scaling all of our core businesses with PostgreSQL. Depending on your dataset, it can be fairly easy to shard your data for a horizontal scale-out (think of independent customer datasets). We have lots of databases that we scale horizontally to much bigger numbers. But, we also developed several tools that makes working with shards mostly transparent. Did not find a better source but one way we use PostgreSQL can be seen in detail in the following slides: http://gotocon.com/berlin-2013/presentation/Why%20Zalando%20...

needusername 154 days ago

5 TB easily fits on a single Oracle instance on a single host.

detaro 154 days ago

It also easily fits in a Postgres instance on a single host, if you only look at "can store X amount of data".

And even if Postgres is slower, for the money you save in license costs you can buy a few beefy nodes extra.

brianwawok 154 days ago

You can easily buy 10 postgres nodes for the license cost of 1

Oracle node.

▲ needusername 153 days ago

And your engineers work for free.

▲ mhd 153 days ago

Free, no. Less money than Oracle DBAs? (Yes.

▲ philsnow 154 days ago

> To edit a Pin, we read-modify-write the JSON under a MySQL transaction:

I've seen a different approach where you keep a version number on the row, do your read, modify in memory on the app server, then do your write like this

```
    UPDATE db03429.pins SET blob=’<modified blob>’ WHERE local_id=7075733 AND version=53
```

then look at the result and make sure that it modified one row. If it returned zero rows, you retry (or show a failure to the user, whichever is appropriate for your use case).

The reason you'd do this is so you can't ever have the row locked for a long period of time. A lot of people don't think about database scalability so _even if they know_ that the code they're writing runs while a transaction is held open, they don't care that that transaction is blocking anything else that is trying to read the row they're working on.

This can lead to row lock bloat over time, which can cause scalability / availability issues as app servers wait longer and longer to get read (or write) locks on all the rows they care about for their current request. This is mitigated a bit if you're requiring / encouraging people to read from slaves instead of master, though.

▲ philsnow 154 days ago

> We only interact with the master in production. You never want to read/write to a slave in production. Slaves lag, which causes strange bugs.

These can be worked through, with discipline. You probably only need a fully consistent few on a surprisingly small number of pages.

Generally you'll want to read from the master right after a user changes something (if you have a model where user clicks a button -> you go to a different page). User changes setting on a pin, clicks 'save', you render a new page showing their updated pin. This page view should probably come from the master, or else you risk the user's change not showing up, causing confusion.

Reads from slaves are fine, as long as you're not using something that was read out of a slave as an input to a database write somewhere else (which you shouldn't be doing anyway!). If you render a page that lets a user change their profile (say site.com/edit_profile), the user data can come from a slave, but if you take _all_ the field values and blindly write those into the master, that's where you run into "time travel" bugs. You just need to find out what the user changed and only make those changes in the master.

▲ misiti3780 154 days ago

Can someone explain to my why people are still recommending mysql over postgres? This is a serious question, it just seems that Postgres has more features and I cant think of any good reasons mysql would scale any differently other than it has been along a little longer (so there are more blog posts + experienced engineers) ?

"MySQL is mature, stable and it just works. Not only do we use it, but it’s also used by plenty of other companies pushing even bigger scale. MySQL supports our need for ordering data requests, selecting certain ranges of data and row-level transactions. It has a hell of a lot more features, but we don’t need or use them. But, MySQL is a single box solution, hence the need to shard our data. Here’s our solution:"

what about that paragraph is not true of postgres also ?

UPDATE:

This schemaless json reminds me of this friendfeed blog post from years ago:

https://backchannel.org/blog/friendfeed-schemaless-mysql

▲ morgo 154 days ago

There are features on both sides which the other database doesn't have.

With this specific workload, I think MySQL will work pretty well. Two features in particular: innodb clustered index and compression.

(I work on the MySQL team.)

▲ misiti3780 154 days ago

thank you!

▲ pbreit 154 days ago

Is it surprising that Pinterest is on MySQL? Has Postgres finally usurped the position of default DB for startups?

▲ schneems 154 days ago

Came here to say the same. It makes sense that older companies are locked into their database as migrating would be too difficult like Facebook. I though Pinterest was a relatively young company and I'm surprised it chose mysql. Instagram is build on top of postgres and was founded in 2010, pinterest was founded in 2009.

▲ taf2 154 days ago

Other than Oracle being a part of the equation MySQL is still an open source database and too quote the article:

"Aside: I still recommend startups avoid the fancy new stuff — try really hard to just use MySQL. Trust me. I have the scars to prove it."

In many ways, Postgres is cutting edge with it's features and capabilities. I see lots of updates from Postgres that include new SQL features (json objects) - conversely I see lots of updates from MySQL that are about durability and scalability e.g. (galera cluster, percona updates etc...)

From the use case Pinterest exposed in the article, I'm not sure what the additional features of Postgres would actually buy them?

▲ forkerenok 154 days ago

Well, that's exactly the thing about Postgres.

It is not stuck in the stone age, yet extremely reliable. Having worked with a number of MySQL (main fork) and Postgres installations in the past 7 years I observed a number of crashes and corruptions with MySQL (not counting the glory MyISAM days) while having _none_ with Postgres.

I should confess that that experience made me very biased in favour of Postgres. My default thinking about Postgres now is: if it doesn't have something MySQL has, it is either for a reason or they are taking their time to do it the _right_ way.

▲ techman9 154 days ago

JSON object storage for one? I'm a little confused as to why their schema relies on storing JSON data as a TEXT field in MySQL. I don't believe this is technically incorrect, but seems to negate many of the features of relational database design. Why not just store each entry as either a traditional column or use a solution that enables native JSON storage.

▲ mappu 154 days ago

>*Why not just store each entry as either a traditional column*

Slow ALTER

>*or use a solution that enables native JSON storage*

This would help if you needed to select or join on the individual columns, but if Pinterest don't need to do that, then this falls under the *"avoid the fancy new stuff"* quote from the article.

▲ techman9 154 days ago

> Slow ALTER

In practice, how often do schema migrations take place though?

▲ mappu 154 days ago

Looking over the source history at $DAYJOB, we run a migration containing DDL statements on average every two days. Mostly for adding new features on small(er) metadata tables where the ALTER doesn't hurt - but occasionally on our bigger XX-GB log tables, where it takes all morning.

▲ rimantas 154 days ago

I believe Facebook had enough talent capable to choose their DBs carefully. And I am pretty sure MySQL replication had a lot to do with it. As having people who already knew how to finetune MySQL for big loads. BTW, believe it or not, Wikipedia once run on Postgresql, but was later migrated to MySQL (and currently they use MariaDB).

▲ forkerenok 154 days ago

Proof-link? or lie. (pg fanboy here)

▲ e12e 154 days ago

I'd also like to see a reference to *wikipedia* running on postgresql. The mediawiki software platform supports running on postgresql[1] -- but it appears the wikipedia has been running on MySQL prior to migrating to MariaDB:

http://blog.wikimedia.org/2013/04/22/wikipedia-adopts-mariad...

[1] https://www.mediawiki.org/wiki/Manual:PostgreSQL

▲ z3t4 154 days ago

Nice article and good explanations.

But I still wonder why some people store blobs in a database rather then simple files on a file-system!?

▲ thezilch 154 days ago

There are a lot of things your FS is not going to give you (eg. transactions), and if you're going to have to query it anyway to get that, you might as well have your blob there too.

For "small" blobs, say <256K, your database is probably faster too. I can only recall one such study (and of only NTFS vs MSSQL), and NTFS was only advantageous at >1MB blobs: http://research.microsoft.com/pubs/64525/tr-2006-45.pdf

▲ taf2 154 days ago

also operationally it's easier... you can rely on replication to backup those files instead of separate scripts that would need to work in-conjunction with the database replication.

If say you have the database with blobs - you can restore the db and everything is back up and running. If instead, you need to also restore the individual files... well - not only do you have lots of individual disk writes, you also have to make sure you still have all the files...

▲ rakoo 152 days ago

Even better from the SQLite page (https://www.sqlite.org/intern-v-extern-blob.html), comparing storing content in SQLite versus storing content in files and filenames in SQLite. The bonus of mixing both is that you get all niceties from a SQL store (transactions, using SQL to query...).

The takeaway is that for content < 20K you should use SQLite, and for content < 100K you should also use it with possibly some tuning to do.

▲ e40 154 days ago

Is this the guy that's going to reddit?

▲ donjigweed 154 days ago

Very nice guy. He used to work at Azul with Cliff Click.

▲ econner 154 days ago

Yes.

▲ P3R3 154 days ago

Yes

▲ fredliu 154 days ago

How are they going to solve the problem of querying into the data that's stored as json? E.g. trying to find all pins whose "link" is from, say, reddit. Just pull out all data and filter them through in client side? That's not gonna scale. Or having a sort of cron job that periodically picking out interested fields in new json data and store them in a separate table? <-- this is essentially what we do in one of our projects, but curious to see how they do it, or alternatives.

numbsafari 154 days ago

I can think of two ways.

The first would be to create a mapping table as described. For a relation like "links to reddit", the cardinality is such that it would probably break their sharding scheme.

So the second approach is probably the one he mentions in the article: map reduce (more generally, separate computation). My guess is that for those sorts of "reports" they are using Hadoop. They could also be leveraging things like the HyperLogLog features of Redis.

morgo 154 days ago

In MySQL 5.7 this will be possible because there is a native JSON data type + indexing available via computed columns.

fredliu 154 days ago

That's good to know. 5.7 seems a bit new though. Before 5.7, what are the common practice to query into json?

fredliu 154 days ago

Not sure why I'm down voted, that was a genuine question...

ck2 154 days ago

PINTEREST: publish your public outgoing IPs so we can whitelist you and block the rest of amazon ec2

It is the responsible thing to do, otherwise other website bots can spoof you.

joshenders 152 days ago

Hi there, Pinterest traffic engineer here. I've added a task to take care of this. Thanks for the suggestion!

ck2 152 days ago

I'm embarrassed I didn't say "please". But thank you!

windowsworkstoo 154 days ago

Agreed, or implement RDNS verification like the major search engines do.

detaro 154 days ago

What's the problem with other bots appearing as pinterest?

boomzilla 154 days ago

Simple economics: they get some value from having their content on Pinterest. There is no value (that they could see) from having their content scraped by other bots.

▲ detaro 154 days ago

If that's the only reason then I hope pinterest doesn't publish their IPs.

▲ mappu 154 days ago

That's terrible for whoever the next big Pinterest is.

▲ mark242 154 days ago

Am I missing the obvious? They're using text columns as blob stores for JSON data? How on earth do you query that in MySQL? How do you run a secondary index, on say the user_id? Is it just one gigantic instance of Elasticsearch or Lucene?

▲ joantune 151 days ago

HBase? no idea, I would also like to know that one.

▲ applecore 154 days ago

This reminds me of how FriendFeed was using MySQL to store unstructured data.

http://backchannel.org/blog/friendfeed-schemaless-mysql

▲ ranyefet 154 days ago

This is exactly what I was thinking

▲ baghali 154 days ago

I'm curious to know how Pinterest or Instagram decide about what goes inside each shard? Do they shard by user ids or something else? Secondly, would like to know if a shard gets more data than other shard, how do they load balance?

▲ joantune 153 days ago

a minor con of this approach is that you have to add an extra layer to your application to do these operations, i.e. to abstract them. But probably the speed & other gains surpass the cons, and in such a big team, i'm sure they will easily handle such an abstraction layer.

One good idea is to open source it, so if other people can take advantage of it, they will also help you maintain it and find bugs for it.

A question: If you started this now, would you consider using Postgres-XL AFAIK it supports similar shardings, in a more transparent manner for the developers. Any thoughts on this?

▲ uptown 154 days ago

Can somebody help me understand this conversion?

Shard ID = (241294492511762325 >> 46) & 0xFFFF = 3429

Type ID = (241294492511762325 >> 36) & 0x3FF = 1

Local ID = (241294492511762325 >> 0) & 0xFFFFFFFF = 7075733

▲ blyxa 154 days ago

bitwise operations.

241294492511762325 >> 46 means shift the binary form of number which is

```
0000001101011001010000000001000000000000001101011111011110010101
```

to the right 46 spaces while replacing with 0s

```
0000000000000000000000000000000000000000000000000000110101100101
```

0xFFFF is hex version of 1111111111111111 (note two missing bits he reserved for gold :D)

& 0xFFFF is doing a AND operation of the bits.

```
0000110101100101
1111111111111111
________________
0000110101100101
```

so 0000110101100101 in decimal is 3429

▲ econner 154 days ago

These are bit shift operations with a bitmask. Basically the ids are 64-bit and sections are reserved to identify the shard, type, and local id.

▲ thewarrior 154 days ago

I know nothing about scaling databases but this reminds of Amazon's Dynamo DB object store but without the consistent hashing trick .

▲ econner 154 days ago

Dynamo is very complex, beyond consistent hashing. It also uses (or used) gossip and lots of tricks (hacks) to gracefully handle adding and removing nodes and distributing data to new nodes, etc. This uses a simple partitioning scheme albeit it does have its own consistency issues.

▲ silverlake 154 days ago

Isn't it unsafe to expose internal database IDs to external clients? I would have generated a second GUID for public view.

▲ brianwawok 154 days ago

Depends on what the GUID lets you access, right?

If GUID got you into a bank account, opps.

If it gets someone access to an already public picture... not a huge deal, right?

▲ wereHamster 154 days ago

But how do you ensure that it is unique?

▲ silverlake 154 days ago

Twitter's Snowflake, though it's gone now. The principle is the same. https://github.com/twitter/snowflake.

▲

▲ stevecalifornia 154 days ago

You will have a mapping table: Public_Guid, Private_ID

Every request has to convert the public, non-guessable GUID into the private ID that will be used in the lookup query.

▲ tapirl 153 days ago

Too many configs and maintenance.

NoSQL is really better than SQL to scale your databases. It is just sad there is no one open source NoSQL db as good as Google BitTable.

▲ ScottWhigham 153 days ago

NoSQL is really better than SQL to scale your databases.

Downvoted for what is essentially your opinion presented as a fact. If you want to make such a statement, you'll need to give more context - either in the form of proof, experience, or "other". Just something would suffice really - anything. You'll find a lot of people agreeing with you if you add such context.

Without context, it's nothing more than "Vanilla is better than chocolate - hands down."

▲ tapirl 149 days ago

NO context, it is just absolutely.

▲ loco5niner 154 days ago

> Shard. Or shard not. There is no try.

sorry, that was bugging me...

▲ pferrel 154 days ago

How embarrassing for them, conquering yesterday's tech today!

▲ halayli 154 days ago

This looks like a big hack to compensate for using the wrong tool. Cassandra would have been a better solution IMO.

With Cassandra, you can set replication factors, speed up the writes, and automatically shard the data without having to manage your own "mapping tables".

▲ econner 154 days ago

Cassandra was immature when this work was started (late 2011). The team was also much more familiar with the ins and outs of operating mysql.

▲ richardwhiuk 154 days ago

There's also no transactions in Cassandra, although that's questionable useful given that the unidirectional map may be on a different machine.

▲ brianwawok 154 days ago

Cassandra has very lightweight transactions.

What he lists as his transaction use case (update if unmodified) I do right now with Cassandra and an IF clause and a timestamp...

i.e. update foo set x = y AND last_modified_timestamp = 456789 IF last_modified_timestamp = 12345

see http://www.datastax.com/dev/blog/lightweight-transactions-in...

▲ halayli 154 days ago

Technical debt is not an excuse to use the wrong tool. This is how you end up having a hammer.

Cassandra was mature by late 2011 and we were using it in production back then (billions of records).

▲ luckycharms810 154 days ago

Technical debt doesn't necessarily arise from using the wrong (a.k.a pretty similar but with a few features less than what you need) tool. In my experience, it's more likely that technical debt arises from trying to use thew newest tool (which looks to be a silver bullet) rather than trusting your own experiences. It's surprising to me that people are willing to move to a completely different technology rather than extend and repurpose what your engineers are already familiar with.

▲ joantune 153 days ago

Although, by using an existing tool, as long as it fits your needs, you'll avoid reinventing the proverbial wheel, ofcourse at the expense of learning something new.. which might not be a bad thing..

▲ econner 154 days ago

The mapping tables are used to store relationships between objects, not to define shard locations. The shard mapping is encoded in the object id.

In any case, I would have a knee jerk reaction to not trust any database system that hit 1.0 a month before I wanted to start using it, especially for mission critical core data.

▲ acveilleux 154 days ago

Doubly so when everything is catching fire around you from how overloaded it all is.

▲ wereHamster 154 days ago

For this very reason many projects nowadays jump directly from 0.x to 2.x :) 2 is the new 1.

▲ _up 154 days ago

But if you also need to use MySql anyway, Cassandra would be the "wrong Tool". Because you would have to maintain two DBMS instead of one. You want to minimize complexity and this can mean implementing your own Sharding.

▲

nemothekid 154 days ago

Even today I'm not sure I'd recommend Cassandra for his use case. A stated requirement he needed was:

>Support asking for N number of Pins in a board in a deterministic order (such as reverse creation time or user specified ordering). Same for Pinner to likes, Pinner to Pins, etc.

This can be a pain to model in Cassandra. It would require denormalizing for every key you wanted to order on, and it generally makes updating data a pain.

jlongtine 154 days ago

I don't think that the schema here allows you to do this kind of query even in MySQL. They are throwing everything in that blob, which makes it pretty difficult to sort on anything that's in the blob. So, I think they would have to do what you are describing, denormalize everything... or get all the data and do the sort in the app. Which is precisely what would be required with Cassandra or another NoSQL system. Am I missing anything that would invalidate what I've just said?

brianwawok 154 days ago

If you only need 1 ordering, not too hard in cassandra. If you need a few different orderings, you would need a side table - which seems to be what you need here with mysql.

meritt 154 days ago

While that would have made scaling easier they would then run into a different set of massive problems because Cassandra isn't a relational database.

brianwawok 154 days ago

But neither is a sharded database. You can't do fancy joins and groups across 400 shards. (Well you can, but you are writing your own code to split up requests and stick it back together.. which is the same thing you would need to do in Cassandra).

georgiecasey 153 days ago

> 'We had several NoSQL technologies, all of which eventually broke catastrophically'

axiak 154 days ago

They also use HBase for a lot of their non relational data.