What is sharding and why is it important?

Asked 15 years, 6 months ago Modified 8 months ago

Viewed 127k times

210

I think I understand sharding to be putting back your sliced up data (the shards) into an easy to deal with aggregate that makes sense in the context. Is this correct?

Update: I guess I am struggling here. In my opinion the application tier should have no business determining where data should be stored. At best it should be shard client of some sort. Both responses answered the what but not the why is it important aspect. What implications does it have outside of the obvious performance gains? Are these gains sufficient to offset the MVC violation? Is sharding mostly important in very large scale applications or does it apply to smaller scale ones?

database terminology

Share

Improve this question

Follow

edited Jun 23, 2009 at 3:44

asked Jun 14, 2009 at 15:09

ojblass
21.6k ● 23 ● 86 ● 136

1 Would one of these webinars be helpful? vimeo.com/26742356 slideshare.net/rightscale/… vimeo.com/32541189 – user1181262 Jan 31, 2012 at 20:27

8 Answers

Sorted by: Highest score (default)

200

Sharding is just another name for "horizontal partitioning" of a database. You might want to search for that term to get it clearer.

From Wikipedia:

> Horizontal partitioning is a design principle whereby rows of a database table are held separately, rather than splitting by columns (as for normalization). Each partition forms part of a shard, which may in turn be located on a separate database server or physical location. The advantage is the number of rows in each table is reduced (this reduces index size, thus improves search performance). If the sharding is based on some real-world aspect of the data (e.g. European customers vs. American customers) then it may be possible to infer the appropriate shard membership easily and automatically, and query only the relevant shard.

Some more information about sharding:

> Firstly, each database server is identical, having the same table structure. Secondly, the data records are logically split up in a sharded database. Unlike the partitioned database, each complete data record exists in only one shard (unless there's mirroring for backup/redundancy) with all CRUD operations performed just in that database. You may not like the terminology used, but this does represent a different way of organizing a logical database into smaller parts.

Update: You wont break MVC. The work of determining the correct shard where to store the data would be transparently done by your data access layer. There you would have to determine the correct shard based on the criteria which you used to shard your database. (As you have to manually shard the database into some different shards based on some concrete aspects of your application.) Then you have to take care when loading and storing the data from/into the database to use the correct shard.

Maybe this example with Java code makes it somewhat clearer (it's about the Hibernate Shards project), how this would work in a real world scenario.

To address the "`why sharding`": It's mainly only for very large scale applications, with *lots* of data. First, it helps minimizing response times for database queries. Second,

you can use more cheaper, "lower-end" machines to host your data on, instead of one big server, which might not suffice anymore.

Share Improve this answer Follow

edited Nov 15, 2013 at 12:07

answered Jun 14, 2009 at 16:53

MicSim

26.8k ● 16 ● 93 ● 136

1 Forgive me but shouldn't the database make the determinations of where to store data. Does this affect code on the application tier? – ojblass Jun 23, 2009 at 3:41

7 I've long been trying to understand how it's different from horizontal partitioning, and the link in your answer kinda proves there's no difference. As someone says in comments to Theo Schlossnagle's post, "...If you are from a traditional database culture your doing horizontal partitioning, if you are from a Web cultur, it is 'Sharding'..." – andreister Nov 15, 2011 at 9:59

@andreister From what I'm reading, sharding is conceptually different in that it's defined by horizontal scaling across multiple logical or physical nodes (in the case of my understanding (mySQL) multiple databases, most likely housed on different logical hardware). Horizontal partitioning is a less specific term, of which "Sharding" is a subset. Again using mySQL as an example, a mySQL partition is handled by a single db instance, which is 100% transparent to the application. A sharding approach would involve either a proxy or an application that intelligently chose which instance. – NateDSaint May 18, 2012 at 14:57

According to wikipedia "Each individual partition is referred to as a shard or database shard." Which is a bit different from the text in the answer that says "Each partition forms part of a shard". – Kevin Wheeler Aug 23, 2015 at 1:19

The wiki article you referenced makes a slight distinction between those two terms. ***Horizontal partitioning*** *splits one or more tables by row, usually within a single instance of a schema and a database server. /***/* ***Sharding*** *goes beyond this: it partitions the problematic table(s) in the same way, but it does this across potentially multiple instances of the schema.* en.wikipedia.org/wiki/… – Peeter Kokk Aug 29, 2016 at 9:18

41

If you have queries to a DBMS for which the locality is quite restricted (say, a user only fires selects with a 'where username = $my_username') it makes sense to put all the usernames starting with A-M on one server and all from N-Z on the other. By this you get near linear scaling for some queries.

Long story short: Sharding is basically the process of distributing tables onto different servers in order to balance the load onto both equally.

Of course, it's so much more complicated in reality. :)

Share Improve this answer

Follow

edited Mar 24, 2018 at 20:19

Nick Coons

3,692 1 20 22

answered Jun 14, 2009 at 15:16

bayer
6,904 • 26 • 35

So sharding affects the design of the data you are storing... sorry if I don't quite understand. – ojblass Jun 14, 2009 at 16:15

Isn't this one horizontal partitioning? – harunurhan Jan 20, 2016 at 20:52

22

Sharding is horizontal(**row wise**) database partitioning as opposed to vertical(**column wise**) partitioning which is *Normalization*. It separates very large databases into smaller, faster and more easily managed parts called data shards. It is a mechanism to achieve distributed systems.

Why do we need distributed systems?

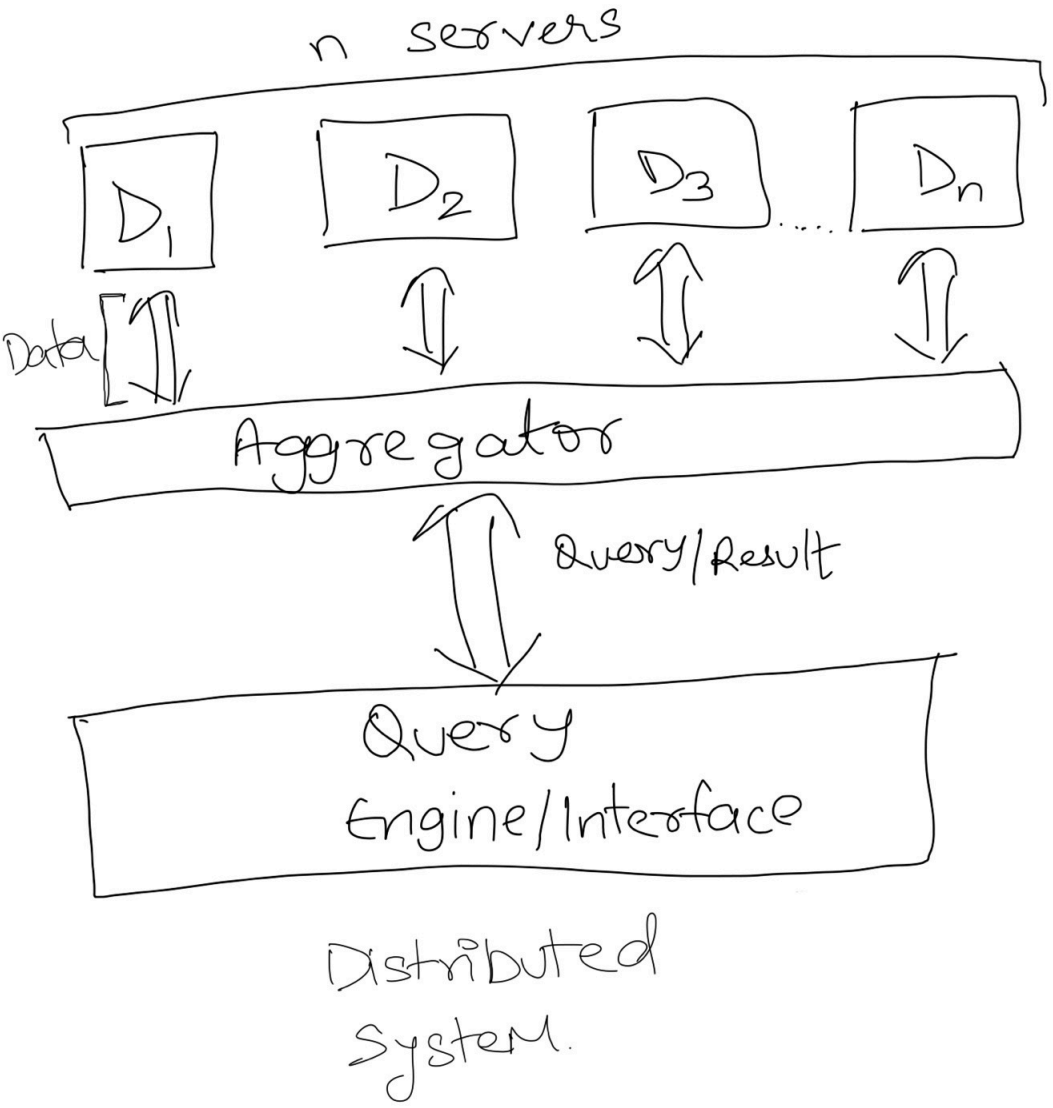
- Increased availablity.
- Easier expansion.
- Economics: It costs less to create a network of smaller computers with the power of single large computer.

You can read more here: Advantages of Distributed database

How sharding help achieve distributed system?

You can partition a search index into N partitions and load each index on a separate server. If you query one server, you will get 1/Nth of the results. So to get complete result set, a typical distributed search system use an **aggregator** that will accumulate results from each server and combine them. An aggregator also distribute query onto each server. This aggregator program is called MapReduce in big data terminology. In other words, Distributed Systems = Sharding + MapReduce (Although there are other things too).

A visual representation below.



Share Improve this answer

Follow

edited Jul 21, 2019 at 14:13

answered Jul 19, 2018 at 10:24

Himanshu Kansal

550 ● 7 ● 18

8

> Is sharding mostly important in very large scale applications or does it apply to smaller scale ones?

Sharding is a concern if and only if your needs scale past what can be served by a single database server. It's a swell tool if you have shardable data and you have incredibly high scalability and performance requirements. I would guess that in my entire 12 years I've been a software professional, I've encountered one situation that could have benefited from sharding. It's an advanced technique with very limited applicability.

Besides, the future is probably going to be something fun and exciting like a massive object "cloud" that erases all potential performance limitations, right? :)

Share Improve this answer

Follow

answered Jun 23, 2009 at 3:54

earino

2,925 ● 21 ● 20

can you share situation where you need sharding
– Gagan Burde Nov 30, 2018 at 10:48

4

Sharding was originally coined by google engineers and you can see it used pretty heavily when writing applications on Google App Engine. Since there are hard limitations on the amount of resource your queries can use and because queries themselves have strict limitations, sharding is not only encouraged but almost enforced by the architecture.

Another place sharding can be used is to reduce contention on data entities. It is especially important when building scalable systems to watch out for those piece of data that are written often because they are always the bottleneck. A good solution is to shard off that specific entity and write to multile copies, then read the total. An example of this "sharded counter wrt GAE: http://code.google.com/appengine/articles/sharding_counters.html

Share Improve this answer

Follow

answered Dec 20, 2010 at 19:17

lampShaded
10.1k ● 2 ● 19 ● 12

9 <<Sharding was originally coined by google engineers>> - not true. Google was founded in 1998. scholar.google.com finds papers from the 1980s like "Discarding obsolete information in a replicated database system" ... The System for Highly Available Replicated Data (SHARD) developed at

CCA... I remember hearing people talking about sharding back then. – Krazy Glew Feb 10, 2018 at 2:14

3

Sharding does more than just horizontal partitioning. According to the wikipedia article,

> Horizontal partitioning splits one or more tables by row, usually within a single instance of a schema and a database server. It may offer an advantage by reducing index size (and thus search effort) provided that there is some obvious, robust, implicit way to identify in which partition a particular row will be found, without first needing to search the index, e.g., the classic example of the 'CustomersEast' and 'CustomersWest' tables, where their zip code already indicates where they will be found.
>
> Sharding goes beyond this: it partitions the problematic table(s) in the same way, but it does this across potentially multiple instances of the schema. The obvious advantage would be that search load for the large partitioned table can now be split across multiple servers (logical or physical), not just multiple indexes on the same logical server.

Also,

> Splitting shards across multiple isolated instances requires more than simple horizontal partitioning. The hoped-for gains in efficiency would be lost, if querying the database required both instances to be queried, just to retrieve a simple dimension table. Beyond partitioning, sharding thus splits large partitionable tables across the servers, while smaller tables are replicated as complete units

Share Improve this answer

Follow

answered Oct 18, 2018 at 10:47

Krishna Rathi

91 • 5

2

> In my opinion the application tier should have no business determining where data should be stored

This is a good rule but like most things not always correct.

When you do your architecture you start with responsibilities and collaborations. Once you determine your functional architecture, you have to balance the non-functional forces.

If one of these non-functional forces is massive scalability, you have to adapt your architecture to cater for

this force even if it means that your data storage abstraction now leaks into your application tier.

Share Improve this answer

Follow

answered Jun 23, 2009 at 4:35

Hans Malherbe

3,018 ● 25 ● 19

2 The application tier can still create separation of data access logic and business rules. This just means you have additional conceptual layers within the "application tier" layer. – Eric Rini Mar 7, 2013 at 17:10

1

Sorry for not going into detail here, but these two articles are the best I've found on sharding and the different strategies as to how this pattern can be implemented.

https://learn.microsoft.com/en-us/azure/architecture/patterns/sharding

> Divide the data store into horizontal partitions or shards. Each shard has the same schema, but holds its own distinct subset of the data. A shard is a data store in its own right (it can contain the data for many entities of different types), running on a server acting as a storage node.

https://www.mongodb.com/features/database-sharding-explained

> Sharding is a form of scaling known as horizontal scaling or scale-out, as additional nodes are brought on to share the load. Horizontal scaling allows for near-limitless scalability to handle big data and intense workloads. In contrast, vertical scaling refers to increasing the power of a single machine or single server through a more powerful CPU, increased RAM, or increased storage capacity.

Share Improve this answer

Follow

answered Aug 31, 2022 at 6:07

Rahul

821 ● 1 ● 13 ● 31