How can Google be so fast?

Asked 16 years, 3 months ago Modified 14 years ago Viewed 86k times

89
votes

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

What are the technologies and programming decisions that make Google able to serve a query so fast?

Every time I search something (one of the several times per day) it always amazes me how they serve the results in near or less than 1 second time. What sort of configuration and algorithms could they have in place that accomplishes this?

Side note: It is kind of overwhelming thinking that even if I was to put a desktop application and use it on my machine probably would not be half as fast as Google. Keep on learning I say.

Here are some of the great answers and pointers provided:

- Google Platform
- Map Reduce
- Algorithms carefully crafted

- Hardware - cluster farms and massive number of cheap computers
- Caching and Load Balancing
- Google File System

performance algorithm

Share

edited Dec 12, 2009 at 21:14
Peter Mortensen
31.6k ● 22 ● 109 ● 133

asked Sep 25, 2008 at 9:42
Jorge Ferreira
97.8k ● 25 ● 126 ● 134

Comments disabled on deleted / locked posts / reviews

19 Answers

Sorted by: Highest score (default)

47 votes

Latency is killed by disk accesses. Hence it's reasonable to believe that all data used to answer queries is kept in memory. This implies thousands of servers, each replicating one of many shards. Therefore the critical path for search is unlikely to hit any of their flagship distributed systems technologies GFS, MapReduce or BigTable. These will be used to process crawler results, crudely.

The handy thing about search is that there's no need to have either strongly consistent results or completely up-to-date data, so Google are not prevented from responding to a query because a more up-to-date search result has become available.

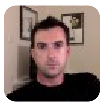
So a possible architecture is quite simple: front end servers process the query, normalising it (possibly by stripping out stop words etc.) then distributing it to whatever subset of replicas owns that part of the query space (an alternative architecture is to split the data up by web pages, so that one of every replica set needs to be contacted for every query). Many, many replicas are probably queried, and the quickest responses win. Each replica has an index mapping queries (or individual query terms) to documents which they can use to look up results in memory very quickly. If different results come back from different sources, the front-end server can rank them as it spits out the html.

Note that this is probably a long way different from what Google actually do - they will have engineered the life out of this system so there may be more caches in strange areas, weird indexes and some kind of funky load-balancing scheme amongst other possible differences.

Share

edited Sep 28, 2008 at 14:54

answered Sep 25, 2008 at 10:09



HenryR

8,509 ● 7 ● 36 ● 39

26 votes

It's a bit too much to put it in one answer.
http://en.wikipedia.org/wiki/Google_platform

Share

answered Sep 25, 2008 at 9:47

Vasil
38k ● 27 ● 93 ● 116

---

22 votes

One fact that I've aways found funny is that Google is in fact run by bioinformatics (’kay, I find that funny because I'm a bioinf…thingy). Let me explain.

Bioinformatics early on had the challenge to search small texts in gigantic strings very fast. For us, the “gigantic string” is of course DNA. Often not a single DNA but a database of several DNAs from different species/individuals. The small texts are proteins or their genetic counterpart, a gene. Most of the first work of computational biologists was restricted to find homologies between genes. This is done to establish the function of newly found genes by noting similarities to genes that are already known.

Now, these DNA strings get very big indeed and (lossy!) search has to be done extremely efficiently. Most of the modern theory of string lookup was thus developed in the context of computational biology.

However, quite some time ago, conventional text search was exhausted. A new approach was needed that allowed searching large strings in sublinear time, that is, without looking at each single character. It was discovered that this can be solved by pre-processing the large string and building a special index data structure over it. Many different such data structures have been proposed. Each have their strengths and weaknesses but there's one that is especially remarkable because it allows a lookup in constant time. Now, in the orders of magnitude in which Google operates this isn't strictly true anymore because load balancing across servers, preprocessing and some other sophisticated stuff has to be taken into account.

But in the essence, the so-called *q-gram index* allows a lookup in constant time. The only disadvantage: The data structure gets ridiculously big. Essentially, to allow for a lookup of strings with up to q characters (hence the name), it requires a table that has one field for each possible combination of q letters (that is, q^S, where S is the size of the alphabet, say 36 (= 26 + 10)). Additionally, there has to be one field for each letter position in the string that was indexed (or in the case of google, for each web site).

To mitigate the sheer size, Google will probably use multiple indices (in fact, they *do*, to offer services like spelling correction). The topmost ones won't work on character level but on word level instead. This reduces q but it makes S infinitely bigger so they will have to use hashing and collision tables to cope with the infinite number of different words.

On the next level, these hashed words will point to other index data structures which, in turn, will hash characters pointing to websites.

Long story short, these *q*-gram index data structures are arguably the most central part of Google's search algorithm. Unfortunately, there are no good non-technical papers explaining how *q*-gram indices work. The only publication that I know that contains a description of how such an index works is … alas, my bachelor thesis.

Share

answered Sep 25, 2008 at 14:14

Konrad Rudolph
545k ● 139 ● 956 ● 1.2k

4 I was in bioinformatics for 5 years, and search engines after that -- and q-grams aren't as important as you think they are. The fundamental data structure for the kind of lookup Google does (on a very, very basic level) is the inverted index. – SquareCog Oct 31, 2008 at 22:31

That seems wrong. Google is running or was running on an inverted index. q-gram will be useful for phrases but not in general – Stefan Savev Feb 1, 2011 at 1:37

@Stefan: The same comment was already made by SquareCog – and I don't deny that inverted indices play a big (and probably much bigger than n-gram indices) role. I picked out this one technology because n-grams are a pet data structure of mine, and I think the key insight – Google is fast because it doesn't actually have to "search", it can do a more or less direct lookup – does depend on such an index (n.b.: this is probably done via hashing but this is *still* an n-gram index). That this index also happens to be inverted is incidental to my point (though probably not for Google ;-)). – Konrad Rudolph Feb 1, 2011 at 7:48

5
votes

Here are some of the great answers and pointers provided:

- [Google Platform]
- [Map Reduce]
- [Algorithms carefully crafted]
- Hardware - cluster farms and massive number of cheap computers
- Caching and Load Balancing
- [Google File System]

Share

answered Jan 6, 2009 at 10:44
Jorge Ferreira
97.8k ● 25 ● 126 ● 134

4
votes

They have implemented good, distributed, algorithms running on a vast amount of hardware.

Share

answered Sep 25, 2008 at 9:49
Anders Sandvig
21k ● 16 ● 61 ● 74

4
votes

One of the most important delays is webservers is getting your query to the webserver, and the response back. THis latency is bound by the speed of light, which even Google has to obey. However, they have datacenters all over the world. As a result, the average distance to any one of them is lower. This keeps the latency down. Sure, the difference is measured in milliseconds, but it matters if the response has to arrive within 1000 milliseconds.

Share

answered Sep 25, 2008 at 9:53

MSalters
179k ● 11 ● 164 ● 368

4
votes

Everyone knows it's because they use pigeons, of course!

Oh yeah, that and Mapreduce.

Share

answered Oct 31, 2008 at 22:18

HanClinto
9,461 ● 3 ● 33 ● 31

If they get rats to work for them, too, two of the most usesless and annoying creatures would have a job... – Xn0vv3r Apr 30, 2009 at 10:59

I laugh a lot with this one haha – victrnava Aug 6, 2009 at 6:24

3
votes

They pretty much have a local copy of the internet cached on thousands of PC's on custom filesystems.

Share

answered Sep 25, 2008 at 9:49

Richard Walton
4,785 ● 3 ● 40 ● 49

Hitting a disk-based filesystem would cost a lot in terms of latency (Amazon found this with Dynamo and sacrificed some resilience for it); I suspect that everything on the critical path is kept in memory. – HenryR Sep 25, 2008 at 10:12

3
votes

Google hires the best of the best. Some of the smartest people in IT work at google. They have virtually infinite money to throw at hardware and engineers.

They use highly optimized storage mechanisms for the tasks that they are performing.

They have geographically located server farms.

Share

answered Sep 25, 2008 at 9:50

Matthew Watson
14.2k ● 9 ● 63 ● 82

3
votes

An attempt at a generalized list (that does not depend on you having access to Google's internal tools):

1. **Parellelize** requests (e.g. break up a single request in to smaller sets)
2. **Async** (make as much asynchronious as possible, e.g. won't block the user's request)
3. **Memory**/cache (Disk I/O is slow, keep as much as possible in memory)
4. **Pre-compute** (Do as much work as possible before hand, don't wait for a user to ask for data/processing)
5. Care about your **front-end HTML** (see Yslow and friends)

Share

edited Oct 31, 2008 at 22:13

answered Oct 31, 2008 at 18:58

Jilles

748 ● 4 ● 11

2

votes

You can find in the google research homepage some pointers about the research papers written by some of the google guys. You should start with the explanatio of the google file system and the map/reduce algorithm to try and understand what's going on behind the google pages.

Share

answered Sep 25, 2008 at 9:51

Nicolas

24.7k ● 5 ● 61 ● 67

2

votes

This link it also very informative Behind the scenes of a google query

Share

answered Nov 2, 2008 at 19:52

user20804

91 ● 2 ● 4

1

vote

Hardware.

Lots and lots of hardware. They use massive clusters of commodity PCs as their server farm.

Share

answered Sep 25, 2008 at 9:44

TraumaPony
10.8k ● 13 ● 57 ● 75

Just to clarify 'massive': hundreds of thousands of servers. I guess none outside Google knows the real number and it must be changing all the time. – Sergio Acosta Sep 25, 2008 at 9:51

1
vote

TraumaPony is right. Tons of servers and smart architecture for load balancing/caching and voila you can run query in under 1 second. There was a lot of articles on the net describing google services architecture. I'm sure you can find them via Google :)

Share

edited Sep 25, 2008 at 10:10

answered Sep 25, 2008 at 9:47

aku
124k ● 33 ● 176 ● 203

1
vote

HenryR is probably correct.

Map Reduce does not play a role for the search itself, but is only used for indexing. Check [this video interview with the Map Reduce inventors]().

Share

answered Oct 8, 2008 at 15:44

kohlerm
2,624 ● 1 ● 18 ● 23

1
vote

An additional reason appears to be that they cheat on the TCP slow start algorithm.

http://blog.benstrong.com/2010/11/google-and-microsoft-cheat-on-slow.html

Share

answered Nov 26, 2010 at 18:29

Thomas Langston
3,763 ● 1 ● 26 ● 41

0
votes

And algorithms that can harness that hardware power. Like mapreduce for instance.

Share

answered Sep 25, 2008 at 9:48

Vinko Vrsalovic
340k ● 55 ● 340 ● 373

MapReduce is not used to respond to queries. – MSalters Sep 25, 2008 at 9:50

MapReduce runs on a large cluster of machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters daily – Vinko Vrsalovic Sep 25, 2008 at 9:56

MapReduce is almost certainly used to asynchronously index crawler data. I'd be very surprised if it were on the critical path for search. Firing off a MapReduce job would really kill latency. – HenryR Sep 25, 2008 at 10:11

Henry -- they might be using it for routing in directions/maps. But yes, to the general case. You don't want any hardcore computation to happen in order to respond to a regular user query. – SquareCog Oct 31, 2008 at 22:34

0

votes

If you are interested in more details about how the google cluster works, I'll suggest this open source implementation of their HDFS.

It's based on Mapreduce by google.

Share

answered Sep 25, 2008 at 9:50

yann.kmm
837 ● 7 ● 21

HDFS is a distributed file system. The mapreduce clone is called Hadoop, and can run either on HDFS or on your local file system. – SquareCog Oct 31, 2008 at 22:22

0

votes

1. Multi staged data storage, processing and retrieval
2. EFFICIENT Distribution (100's of 1000's of machines) of the above tasks
3. Good framework to store the raw data and the processed results
4. Good framework to retrieve the results

How exactly all these are done is summarized by all the links that you have in the question summary

Share

answered Sep 25, 2008 at 16:57

computinglife

4,391 ● 1 ● 23 ● 18