System Design for Facebook

Note: This is just a brainstorming article.
It could be a complete mismatch with what actually goes on in Facebook.
This only shows how one can logically reason and try to draw some estimates about a big cloud service.

According to this reference: Facebook stats on Zephoria, Facebook handles (as of 2013-2014):

1. 5 billion likes daily
2. 1 billion logon daily
3. 5 new profiles every second
4. 0.5 billion photos uploaded per day

Lets see how this translates into system load per second.
(Assuming 10^5 seconds per day instead of the actual 86,400)

Type	Requests Per day	Requests Per second	Assumptions	Data size Per second
Likes	5x10^9	50,000	Assuming 500 bytes per request	500 x 5x10^4bytes = 25x10^6bytes = 25 MB/sec
Photo uploads	0.5x10^9	5,000	Assuming 1 MB per photo	5,000 MB / sec = 5GB/sec
Comments (Assuming 40% of likes)	2x10^9	20,000	Assuming 2 kb per comment	2 x 2x10^4kb/sec = 40 MB/sec
Totals		75,000		~ 5.1 GB/sec

Estimation of storage

5.1 GB/sec translates to 5.1 x 60 x 60 x 24 x 365 = 160 PB
Assuming Facebook keeps past 3 years of data for serving promptly (and data before that is
accessed rarely from a different kind of storage), we need a storage size of around 500 PB.

This is about one-third the size of database we calculated for Google Maps and so should be manageable.

Number of storage nodes required

Assuming a single machine could manage around 15 TB of space, we would need 500 PB/ 15 TB = 30,000 database shards.
We would need some replicas with each shard to handle read traffic and to add fail-tolerance.
Assuming 1 master 2 replicas per shard system, we would need 100,000 database nodes , each holding 15 TB of space.

Note that this kind of data is perfect for caching as once uploaded, a photo will never be changed.
(It can be deleted, but not updated).
Comments and likes can be updated but that is also not very frequent and we can omit those updates here.
So instead of this giant cloud of DB-nodes, we could replace those

Traffic analysis for Facebook

We got around 75,000 requests per second above for facebook.
But those are only write queries. Number of reads would be substantially higher.
Assuming a factor of 10 reads per writes, we get around 750,000 requests per second for read.

Here are two sources to find maximum queries per second a server can handle:

1. Server Performance at Dreamhost wiki
2. Our precious StackOverflow

These links talk about 10,000 to 15,000 read requests per second if most of the data is from cache.
For our case, let us assume that a single server can handle 10,000 reads per second and

1% of write load (i.e. 100 writes per second).
Writes are expensive because they need to go through DB constraints, update indexes and caches etc.
So to handle 75,000 writes, we need 75,000 / 100 = 750 shards where each master takes 100 writes per second.
Since the data is distributed among the shards, each shard will receive an equal part of read traffic as well.
That figure will be: 750,000 / 750 = 1000 reads per shard per second which is easily manageable by a single shard-machine as well.
Thus a cloud with 750 shards should be sufficient for this purpose where each machine takes 100 writes/sec and 1000 reads/second.

Also see: Whos got the most web servers