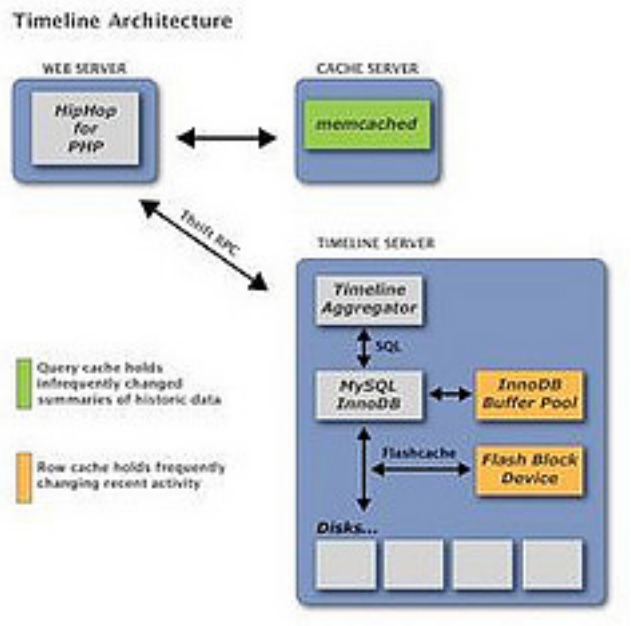
Facebook Timeline: Brought to You by the Power of Denormalization

Monday, January 23, 2012 at 9:14AM

Todd Hoff in Strategy, facebook



Facebook Timeline is audacious in scope. It wants to compile a complete scrollable version of your life story from photos, locations, videos, status updates, and everything you do. That could be many decades of data (hopefully) that must stored and made quickly available at any point in time. A huge technical challenge, even for Facebook, which we know are experts in handing big data. And they built it all in 6 months.

Facebook's Ryan Mack shares quite a bit of Timeline's own implementation story in his excellent article: Building Timeline: Scaling up to hold your life story.

Five big takeaways from the article are:

- **Leverage infrastructure rather than build something new**. You might expect that they would pioneer a new infrastructure for Timeline, but given the short schedule, they decided to go with proven technologies inside Facebook: MySQL, Multifeed (a custom distributed system which takes the tens of thousands of updates from friends and picks the most relevant), Thrift, Memcached,

Operations. The last point about the operations infrastructure is a huge win for any team. All that just works. They can concentrate on solving the problem and skip the whole tooling dance, which is why new products can be generated amazingly fast inside a "big company", if the infrastructure is done right.

- **Denormalize. Format data in the way you need to use it**.
  - Denormalzation, creating special purpose objects instead of distributed rows that must be joined, minimizes random IO by reducing the number of trips to the database. Caching can often get around the need for denormalization, but given the amount of timeline data and how much of it is cold, that is it will rarely be viewed, caching everything isn't a good design.
  - Timeline decides the order to display data by calculating a rank based on metadata. The denormalization process brought all that metadata together in a format that meant ranking could be done in a few IOs and streamed efficiently from the database using a primary key range query
  - Timeline is like a datamart in a data warehouse. Data must be slurped up from dozens of different systems, cleaned, merged, and reformatted into a new canonical format. Facebook of course did this in a Facebook-like way. They created a custom data conversion language, they deployed hundreds of MySQL servers to extract the data out of "legacy" systems as fast as possible, they deployed flash storage to speed up joins, they created a parallelizing query proxy, and they standardized on the Multifeed data format for future flexibility.
- **Keep different types of caches**.
  - **Short term cache.** A timeline of recent activity is frequently invalidated because it is changing all the time as you perform actions through your life. This cache is an in RAM row cache inside InnoDB that uses the Flashcache kernel driver *to expand the OS cache onto a flash device*.

    - **Long term cache**. A query cache is kept in memcached. The results of large queries, like the ranking of all your activities in 2010, can be efficiently cached since they will rarely be invalidated.
- **Run operations locally**. The Timeline Aggregator (geographically clustering nearby check-ins, ranking status updates, etc) runs on each database so it can max out the disks. Only data that needs to be displayed is sent over the network.
- **Parallelize development**. The short 6 month development time was partly a product of the quality infrastructure, but of also running significant development activities in parallel. The development team was split into design, front-end engineering, infrastructure engineering, and data migrations. In parallel they built: UI prototypes on a test backend, production UI on a simulated backend, the scalable backend, the denormalization framework, the data warehouse, and simulated load testing.

Related Articles

- Under the Hood: Mobile Timeline
