Inside Facebook Messages' Application Server

By Jiakai Liu on Thursday, April 28, 2011 at 4:38pm

Facebook Messages represents one of the most technically challenging products that we've ever created. As we mentioned when we launched Facebook Messages, we needed to build a dedicated application server to manage its infrastructure.

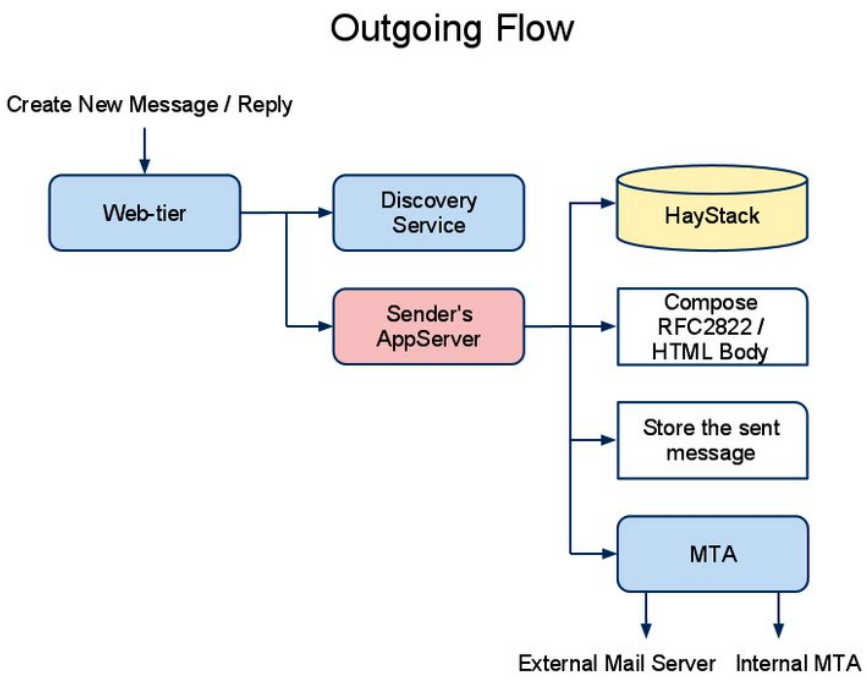
We recently discussed the Messages back end and how we scaled the service to handle all the communication coming from email, SMS, Facebook Chat, and the Inbox. Today we'll explore the internals of the application server.

The Application Server Business Logic

The application server integrates with many Facebook services and shields this complexity from these various clients. It provides a simple interface for its clients to perform standard message operations, including creating, reading, deleting, and updating messages and the Inbox.

The flow for each of them is as follows.

When creating new messages or replying to existing ones, the application server delivers the message to the recipients on behalf of the sending user. If a recipient is specified by his or her email address, the server fetches the attachment from HayStack (if any), constructs the HTML body, and builds an RFC2822 message.

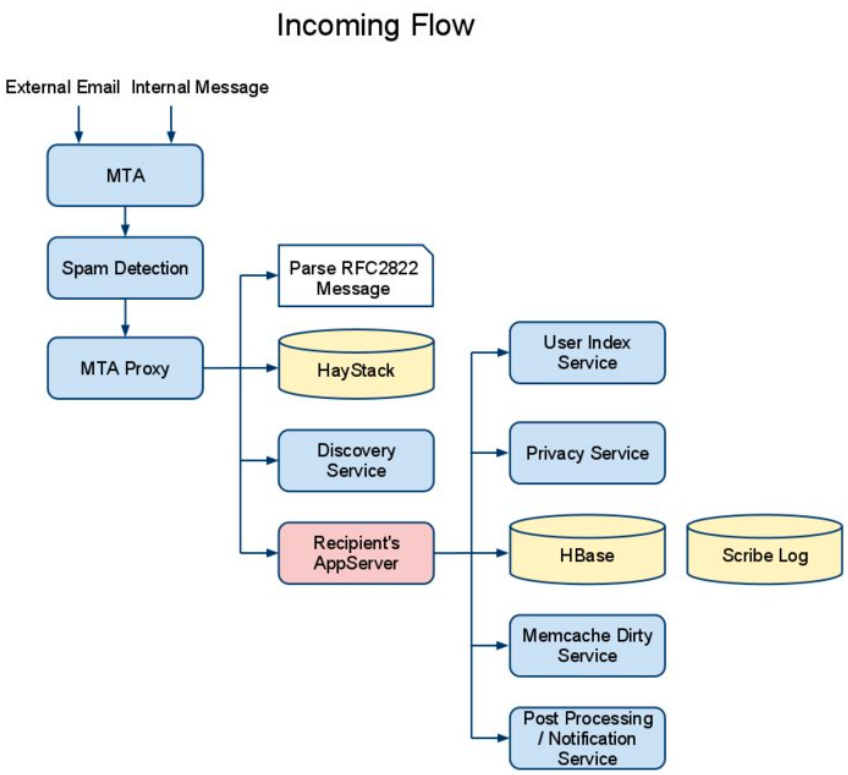


When messages are sent to a user, the server receives them from external email addresses and dispatches incoming messages to the proper recipients if the address is a reply handler. The server finally delivers the message to the user's mailbox, running all pre- and post-processing as needed, and determining the folder and thread where the message should be routed based on a number of signals.

Facebook Engineering

Notes by Facebook Engineering
All Notes

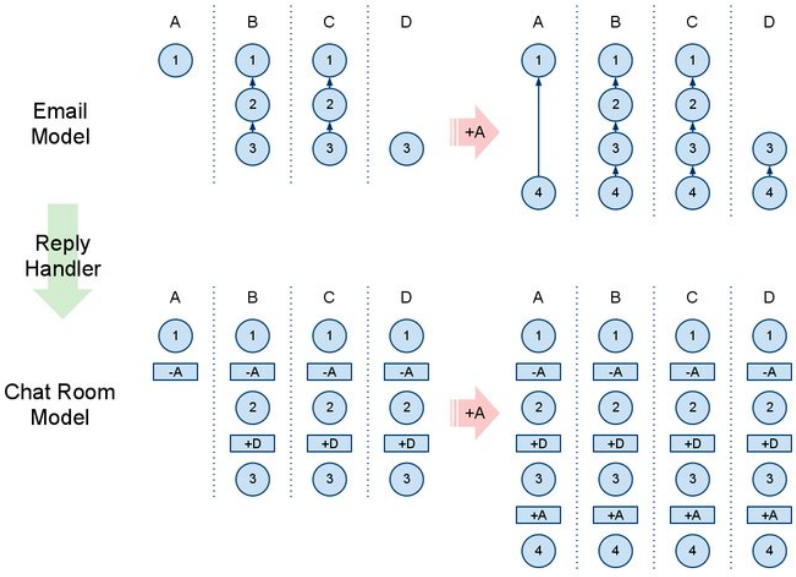
Get Notes via RSS
Embed Post



When reading messages, the server gets various statistics about the user's mailbox, like its capacity; number of messages, threads and replies; and the number of friends with whom the user has interacted. It also gets folder statistics and attributes, the list of threads by various search criteria (folder, attributes, authors, keywords, and so forth), and thread attributes and the individual messages in the thread.

When deleting messages, the server marks messages and threads as deleted. An offline job actually removes the message contents.

When updating messages and threads, the server changes the message- or thread-level attributes, like its read and archive statuses, any tags, and so forth. It also handles subscribe and unsubscribe requests on threads with multiple users.



Email Model vs. Chat Room Model
1. A leaves; 2. D is added; 3. A is added back

Managing Group Threads

Facebook Messages manages group message threads using a chat room model. Users can be added (subscribed) to and leave (unsubscribe from) threads. To enforce this model when email addresses are specified for recipients in a thread, the application server creates a reply handler, like a chat room ID. When an email recipient replies to a thread, the message is sent to the reply handler address.

To optimize read performance and simplify migration and backup processes, message threads are stored with a denormalized schema, so each user has his or her own copy of thread metadata and messages. The server broadcasts subscription and unsubscribe events, synchronizing the thread metadata among all recipients so it can handle the subscription and reply handler in a decentralized manner. The server also manages various corner cases when interacting with users who still have the old Inbox or were subscribed by their email addresses.

Caching User Metadata

When a user access his or her Inbox, the application server loads the most common user metadata (called *active metadata*) and stores it in a least recently used cache. Subsequent requests from the same user can be served promptly with fewer HBase queries.

We need to make fewer HBase queries because HBase doesn't support *join*. To serve one read request, the server may need to look up multiple indexes and fetch metadata and the message body in separate HBase queries. HBase is optimized for writes rather than reads, and user behavior usually has good temporal and spatial locality, so the cache helps to solve this problem and improve performance.

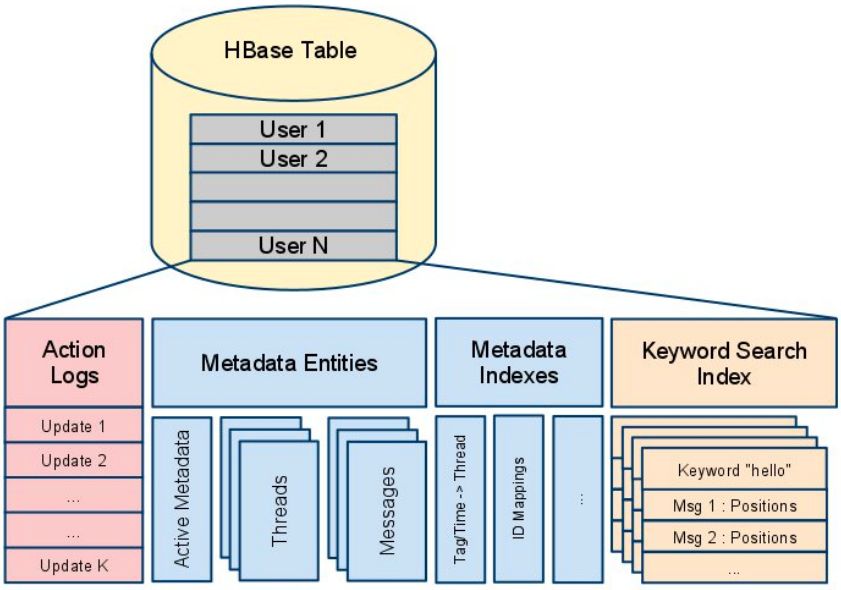
We've also put a lot of effort into improving cache efficiency by reducing the user memory footprint and moving to finer-grained schema. We can cache 5%-10% of our users and have an active metadata cache hit rate of around 95%. We cache some extremely frequently accessed data (like unread message counts displayed on the Facebook home page) in the global memcache tier. The application server dirties the cache when new messages arrive.

Synchronization

HBase has limited support for transaction isolation. Multiple updates against the same user might occur simultaneously. To solve potential conflicts between them, we use the application server as the synchronization point for user requests. A user is served by a particular application server at any given time. This way, requests against the same user can be synchronized and executed in a completely isolated fashion on the application server.

Storage Schema

MTA Proxy strips attachments and large message bodies and stores them in Haystack before they can reach the application server. However, metadata, including search index data and small message bodies, are stored in HBase and maintained by the application server. Every user's mailbox is independent of every other user's; user data is not shared in HBase. A user's data is stored in a single row in HBase, which consists of the following parts:



Metadata Entities and Indexes

Metadata entities contain the attributes of mailbox objects, like folders, threads, messages, and so forth. Each entity is stored in its own HBase column. Unlike a traditional RDBMS, HBase does not have native support for indexes. We maintain the secondary indexes at the application level, where they are stored as key/value pairs in separate columns as well.

For example, to answer the query "loading unread threads on the second page of the Other folder," the application server first looks up the metadata index to get the list of threads that meet the criteria, then fetches the metadata entities of the specified threads, and constructs the response with their attributes.

As we mentioned earlier, caching and effective preloading reduces the number of HBase queries for better performance.

Action Logs

Any update to a user's mailbox (like creating or deleting messages, marking threads as read, and so forth) is immediately appended to a column family in chronological order, called an *action log*. Small message bodies are also stored in action logs.

We can construct or reinstate the current state of user's mailbox by replaying action logs. We use the ID of last action log replayed as the version of metadata entities and indexes. When a user's mailbox is loaded, the application server compares the metadata version and latest action log ID, and updates the mailbox content if the metadata version lags behind.

Storing the action logs at the application level has brought great flexibility:

- We can seamlessly switch to a new schema by replaying the action logs and generating new metadata entities and indexes with an offline MapReduce job or online by the application server itself.
- We can perform large HBase writes asynchronously in batches to save on network bandwidth and reduce HBase compaction cost.
- It is a standard protocol to exchange persistence data with other components. For example, we do application-level backup by writing the action logs to a Scribe log. The migration pipeline converts users' old Inbox data into action logs and generates metadata and indexes with offline MapReduce.

Search Indexing

To support full text search, we maintain a reverse index from keywords to matched

messages. When a new message arrives, we use Apache Lucene to parse and convert it into (keyword, message ID, positions) tuples, then add them to an HBase column family incrementally. Each keyword has its own column. All messages, including chat history, email, and SMS, are indexed in real time.

Testing via a Dark Launch

The application server is new software we built from scratch, so we need to monitor its performance, reliability and scalability before we roll it out to more than 500 million users. We initially developed a stress test robot to generate fake requests, but we found that the results could be affected by quite a few factors, like message size, distribution of different types of requests, distribution of user activity rates, and so forth.

In order to simulate the real production load, we did a *dark launch*, where we mirrored live traffic from Chat and the existing Inbox into a test cluster for about 10% of our users. Dark launches help us uncover many performance issues and identify bottlenecks. We also used it as a convincing metric to evaluate various improvements we made. Over time, we'll continue to roll out the new Messages system to all our users.

Jiakai is a software engineer on the Facebook Messages team.

Alokji Karna FANTASTIC INVENTION BY FACEBOOK. EXCITED TO HAVE THE APPLICATION BROUGHT UP .
April 28, 2011 at 10:16pm

Venkat Pathy And its gonna stink unless u create a db separately fa chat and mail: xxxxx@fb.com
April 28, 2011 at 11:19pm

Erin Ní Chonaill Switch off the fossil fuels and turn on the future! Love FB but do not like the thought of destroying our future by using it.
April 29, 2011 at 12:51am

Tobias J-a It will be up soon, thank you
April 29, 2011 at 4:14am

Vipin Dhiman thanx for posting such a knowledgeable thing...
April 29, 2011 at 6:59am

Paolo Ghezzi Hi i have an idea to propose at facebook can i know how can i do it?
April 29, 2011 at 10:43am · 2

Sade Santangelo Make it like Googlewave. Although all messages would not be able to be deleted (which is kind of nice!), make the 'draft' parts of the message only visible through a 'play' button... after editing the sent message, at first glance only the final messag... See More
April 29, 2011 at 4:45pm · 1

Liyin Tang awesome work !!!
April 30, 2011 at 1:22am

Tushar Pandey thanx
May 1, 2011 at 6:11am

Greg Rose Apache HBase is a great read/write realtime database for hosting huge tables. No join support stinks as well as limited transaction isolation but it is open source so be patient. Caching user metadata is improving efficiency. I am not familiar with Dark launch but it seems like a great testing tool. I am looking forward to seeing the message system evolve in the future.
May 5, 2011 at 9:21pm

Ravi Teja K very nice presentaion.... a lot to know about facebook
May 13, 2011 at 10:57pm

Salma Nehme Get 10 fbml table & All code with video learn,Join now this page = > http://www.facebook.com/Static.FBMLs
May 16, 2011 at 5:05pm

Jaime Maldonado My INBOX chance by MESSAGE and this not work...Can you help me? THANK
May 29, 2011 at 5:12pm

Oliver Beattie It'd be great if it didn't lose messages all the fucking time. I can see people have sent me stuff by the preview line in the Inbox but when I actually go to the thread, missing! Absolutely infuriating.
June 14, 2011 at 11:04am

Bluedavy Lin Very good design,It used all products properly!
June 28, 2011 at 3:05am

Malvinder Singh Dhillon Thats good, but the new chat system have problems. we cant all our online friends all ot once like we used too.. many ppl dont like the new one. is there a way that i dont know of ?
July 21, 2011 at 6:06am

Sung W Cha Message-ID: <OF2E9F5282.012C6F94-ON862579B1.004BB5D3-852579B1.004C213E@mutualofomaha.com>
February 28, 2012 at 2:37pm

Sung W Cha Reply-To: Reply All <100002852883375_0f15825e44e130e55b4798963ae9f5fb@replyhandler.facebook.com>
February 28, 2012 at 2:37pm

Irfan Kanwal fafebook
July 30, 2013 at 7:26pm

Raihan Ahmed What's the name of the App Server that Facebook uses?
October 16, 2013 at 4:09pm

Karthik Srirangam wow. .superb. .Great
December 22, 2013 at 12:06am

Presly Ben Very good design.
December 23, 2013 at 9:36am

Jung Hana I deleted my conversation with my friend by mistake and I want it baaack ..
March 6, 2014 at 11:19am · Edited

Jessie Barber oh and dont evne get me started im suprised someone actully messeged me back about the ,atter but i told the same thing ive been told for ages i even have prof fof my identity i even knwo the email of the acocunt which is still deacativated but is ont eh site iw ant my account bank
August 11, 2014 at 11:41pm

James Godspeed Cannon my facebook server isn't wolking for me,why is facebook doing me this way when i am on facebook.
April 19 at 4:37am