Facebook Chat

By Eugene Letuchy on Tuesday, May 13, 2008 at 10:56pm

One of the things I like most about working at Facebook is the ability to launch products that are (almost) immediately used by millions of people. Unlike a three-guys-in-a-garage startup, we don't have the luxury of scaling out infrastructure to keep pace with user growth; when your feature's userbase will go from 0 to 70 million practically overnight, scalability has to be baked in from the start. The project I'm currently working on, Facebook Chat, offered a nice set of software engineering challenges:

Real-time presence notification:

The most resource-intensive operation performed in a chat system is not sending messages. It is rather keeping each online user aware of the online-idle-offline states of their friends, so that conversations can begin.

The naive implementation of sending a notification to all friends whenever a user comes online or goes offline has a worst case cost of O(average friendlist size * peak users * churn rate) messages/second, where churn rate is the frequency with which users come online and go offline, in events/second. This is wildly inefficient to the point of being untenable, given that the average number of friends per user is measured in the hundreds, and the number of concurrent users during peak site usage is on the order of several millions.

Surfacing connected users' idleness greatly enhances the chat user experience but further compounds the problem of keeping presence information up-to-date. Each Facebook Chat user now needs to be notified whenever one of his/her friends
(a) takes an action such as sending a chat message or loads a Facebook page (if tracking idleness via a last-active timestamp) or
(b) transitions between idleness states (if representing idleness as a state machine with states like "idle-for-1-minute", "idle-for-2-minutes", "idle-for-5-minutes", "idle-for-10-minutes", etc.).
Note that approach (a) changes the sending a chat message / loading a Facebook page from a one-to-one communication into a multicast to all online friends, while approach (b) ensures that users who are neither chatting nor browsing Facebook are nonetheless generating server load.

Real-time messaging:

Another challenge is ensuring the timely delivery of the messages themselves. The method we chose to get text from one user to another involves loading an iframe on each Facebook page, and having that iframe's Javascript make an HTTP GET request over a persistent connection that doesn't return until the server has data for the client. The request gets reestablished if it's interrupted or times out. This isn't by any means a new technique: it's a variation of Comet, specifically XHR long polling, and/or BOSH.

Having a large-number of long-running concurrent requests makes the **A**pache part of the standard LAMP stack a dubious implementation choice. Even without accounting for the sizeable overhead of spawning an OS process that, on average, twiddles its thumbs for a minute before reporting that no one has sent the user a message, the waiting time could be spent servicing 60-some requests for regular Facebook pages. The result of running out of Apache processes over the entire Facebook web tier is not pretty, nor is the dynamic configuration of the Apache process limits enjoyable.

Distribution, Isolation, and Failover:

Fault tolerance is a desirable characteristic of any big system: if an error happens, the system should try its best to recover without human intervention before giving up and informing the user. The results of inevitable programming bugs, hardware failures, et al.,

should be hidden from the user as much as possible and isolated from the rest of the system.

The way this is typically accomplished in a web application is by separating the model and the view: data is persisted in a database (perhaps with a separate in-memory cache), with each short-lived request retrieving only the parts relevant to that request. Because the data is persisted, a failed read request can be re-attempted. Cache misses and database failure can be detected by the non-database layers and either reported to the user or worked around using replication.

While this architecture works pretty well in general, it isn't as successful in a chat application due to the high volume of long-lived requests, the non-relational nature of the data involved, and the statefulness of each request.

For Facebook Chat, we rolled our own subsystem for logging chat messages (in C++) as well as an epoll-driven web server (in Erlang) that holds online users' conversations in-memory and serves the long-polled HTTP requests. Both subsystems are clustered and partitioned for reliability and efficient failover. Why Erlang? In short, because the problem domain fits Erlang like a glove. Erlang is a functional concurrency-oriented language with extremely low-weight user-space "processes", share-nothing message-passing semantics, built-in distribution, and a "crash and recover" philosophy proven by two decades of deployment on large soft-realtime production systems.

Glueing with Thrift:

Despite those advantages, using Erlang for a component of Facebook Chat had a downside: that component needed to communicate with the other parts of the system. Glueing together PHP, Javascript, Erlang, and C++ is not a trivial matter. Fortunately, we have Thrift. Thrift translates a service description into the RPC glue code necessary for making cross-language calls (marshalling arguments and responses over the wire) and has templates for servers and clients. Since going open source a year ago (we had the gall to release it on April Fool's Day, 2007), the Thrift project has steadily grown and improved (with multiple iterations on the Erlang binding). Having Thrift available freed us to split up the problem of building a chat system and use the best available tool to approach each sub-problem.

Ramping up:

The secret for going from zero to seventy million users overnight is to avoid doing it all in one fell swoop. We chose to simulate the impact of many real users hitting many machines by means of a "dark launch" period in which Facebook pages would make connections to the chat servers, query for presence information and simulate message sends without a single UI element drawn on the page. With the "dark launch" bugs fixed, we hope that you enjoy Facebook Chat now that the UI lights have been turned on.

Eugene is a Facebook Engineer

Vikas Malhotra anybody here
April 6 at 10:36am

Vikas Malhotra chat with me
April 6 at 10:36am

Vikas Malhotra sunny where r u
April 6 at 10:48am

Toumi Abdelmalek Bsr
See Translation
April 6 at 3:07pm

Gurpreet Garry hay
April 7 at 11:10pm · 1

Mérlya Angels Hi people!!
April 8 at 10:01am · 1

Jhijhit Sardar Gøød ßøy nice
April 8 at 10:25am

Ashok Kumar Gulani Hi
April 8 at 11:42am · 2

Ganesh Singh Hi
April 8 at 8:56pm

Abduljebbar Idris hiii
April 9 at 2:57pm

Niwemfura Manzi Innocente Amazing!
April 10 at 2:20am

Txuller Zavaly coolll
See Translation
April 10 at 1:32pm

Awis Awis Awais
April 10 at 11:43pm

Sajin Yana da
April 11 at 5:08am · 1

Khanderao Bidve Hi
April 11 at 7:58am

Malvika Chauhan hloo
April 11 at 8:58am · 1

Samuel Akosah Hello
April 11 at 11:37am

Nilmini Wickramanayake Hi
April 11 at 8:08pm · 2

Bayrmaa Bayrhuu hi
April 12 at 4:47am · 1

Cuko Arlindo Hi
April 12 at 10:58am · 2

Sharif Pattan Hi
April 14 at 6:18am · 1

Mina Pari Super.
April 14 at 8:51pm · 2

Hem Kumar Rai Hello happy new year 2072 for all of you
April 15 at 1:12am

Ernst Gaston nice face good
April 16 at 1:16am

Michel Steve Zua How i download it
April 17 at 6:11am

Mujahid Raza HELLO....?
April 18 at 12:20am

Dipen Sharma hi
April 18 at 5:18am

Cecinio Edson Henrique COMEKE
April 18 at 7:18am

Tejkumar Rokhde Hiiii
April 18 at 10:06am

Lino Sarmento Dji
See Translation
April 19 at 12:17pm

Grace de Haro I have a question how can I delete a message that I posted and other messages or pictures as well, is this possible?
April 20 at 6:59am

Sahu Annu bamardhi wat r u doing
April 20 at 10:32am · 1

Suneet Saxena hello how are you
April 21 at 10:27am

Raj Sangada Hi
April 22 at 12:48am

Nhu Tran A
See Translation
April 22 at 4:49am

Ritu Sharma Ho
April 22 at 9:22am

Kartik Rawat Komal jha sexy chat
April 22 at 10:13am

Yassine El Mahi rga3 ba3 lgra3
See Translation
23 hrs

Bianca Gupa hellow
19 hrs

Didier Niyo Azalea Madiba Helo
16 hrs

Isac Ferreira Isac ferreira
See Translation
13 hrs

Mohit Panwar hello
3 hrs