Openfire Scalability

Openfire is a real-time collaboration (RTC) server dual-licensed under the Open Source GPL and commercially. It uses the only widely adopted open protocol for instant messaging, XMPP (also called Jabber). Openfire is easy to set up and administer, but offers rock-solid security and performance.

This document gives an overview of how the Openfire team recently increased the server's scalability by reworking the server's networking layer as well as by optimizing existing code. The details given here reflect tests on a modest deployment; for example, connection managers were not used. Tests on a framework that's closer to real-world usage promise to show even more dramatic improvements.

Summary

In early tests, Openfire developers have demonstrated a server scalability improvement from an approximate 6,000-user maximum in version 3.1.1 to more than 50,000 concurrent users in version 3.2. They achieved these improvements through enhancements in which they:

- Replaced the networking layer with Apache MINA, an open source networking framework that provided support for asynchronous I/O and a foundation for better scaling. Through MINA, Openfire server and connection managers make more efficient use of threads.
- Optimized code to reduce use of performance-expensive APIs and remove unnecessary processing (such as superfluous user validation and XML parsing).
- Implemented a cache to reduce database queries. Administrators can view cache usage data from the Openfire admin console.

For business cases that involve extremely high I/O use (such as many group chats, file transfers, and so on), you will want to use one or more connection managers. For other cases, simply using Openfire should meet the need.

Configuration

Openfire server was deployed to a machine running Sun 280R Server with two 1.2GHz UltraSPARC-III CPUs, 4GB RAM, fiberchannel disks, and FastEthernet (100Mbit/s). A MySQL database was used for stored data. An Openfire plugin generated users, populated 40 rosters with 40 contacts in each, and created vCards. Connection managers were not used; requests were sent directly from simulated users to Openfire server. Memory assigned to the Java virtual machine was 2GB; the server consumed 1GB.

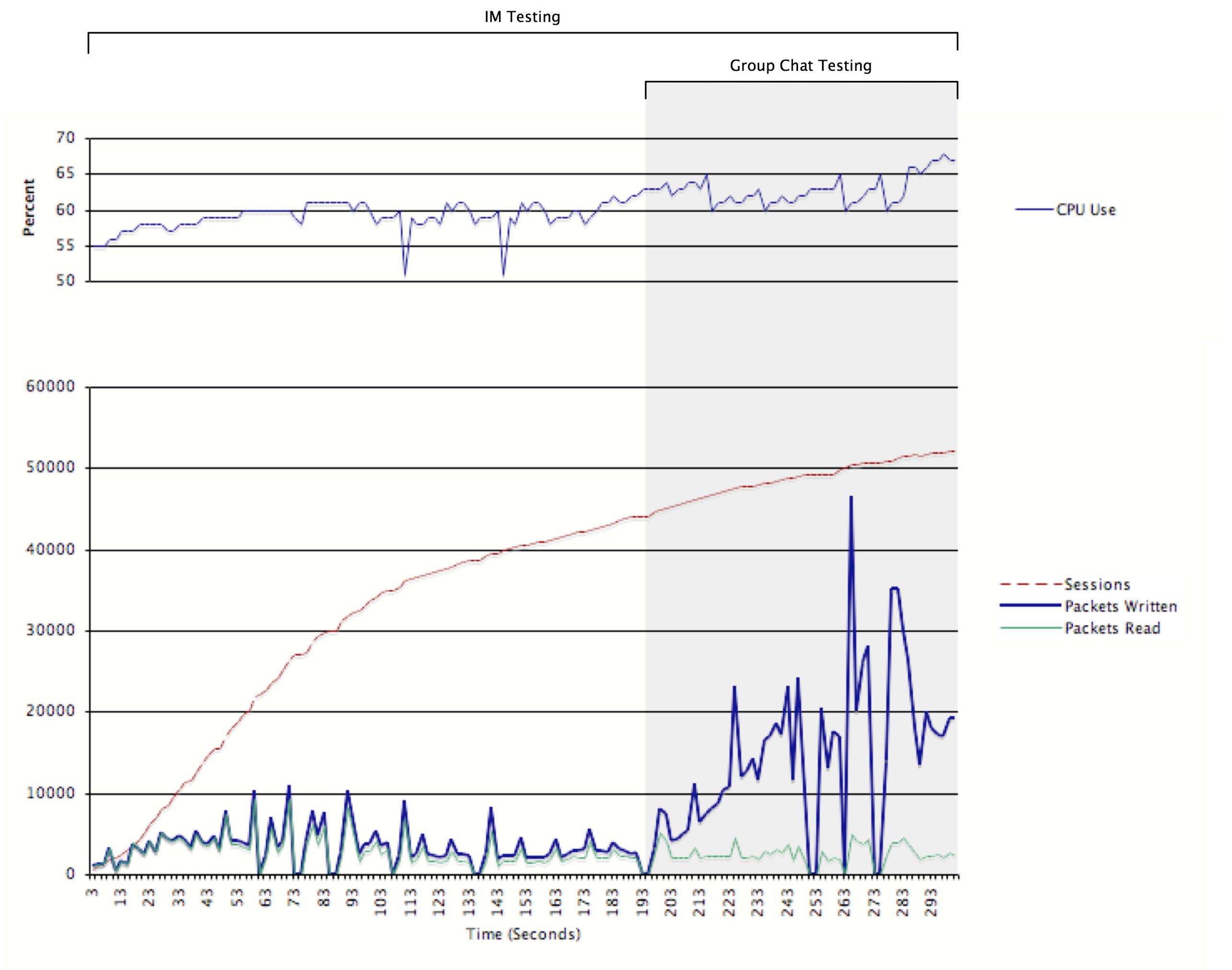
IM tests were run using Tsung, software that simulates users. Tsung was run on a configuration that included two master machines with three slave machines each. The master machines instructed the slave machines to submit client requests to the Openfire server machine. These requests included logging in 500 - 800 virtual users per second. The data requests included authenticating, getting the roster, sending chat messages, and getting user vCards.

Two sets of load tests were used: one to log in a large number of users with low levels of activity, and another to generate a fixed load of 1,500 users who are extremely active and performing resource intensive actions. When one test was failing to add more users to the system, a second test was launched from two different locations to generate additional load of 3,000 extremely active users. Openfire was easily able to handle the additional load generated by these users' activities.

Test Results

In tests simulating 300 to more than 50,000 users, CPU usage increased only gradually for Openfire server and the MySQL database used for Openfire data.

The following illustration shows performance over a period of seconds. Toward the last third of the test, group chat testing was activated to increase load. The bottom portion of the chart shows packet reads and writes as the number of user sessions increases (packet read/writes occur for exchanges of XMPP stanzas such as IM messages, presence notifications, and information queries). The chart's top portion shows percentage of server CPU use for the same period of the test.


IM Testing
Group Chat Testing
Percent
CPU Use
Sessions
Packets Written
Packets Read
Time (Seconds)