PlentyOfFish Architecture

Friday, June 26, 2009 at 3:58PM

Todd Hoff in .Net, Example, Windows

Update 5: PlentyOfFish Update - 6 Billion Pageviews And 32 Billion Images A Month
Update 4: Jeff Atwood costs out Markus' scale up approach against a scale out approach and finds scale up wanting. The discussion in the comments is as interesting as the article. My guess is Markus doesn't want to rewrite his software to work across a scale out cluster so even if it's more expensive scale up works better for his needs.
Update 3: POF now has 200 million images and serves 10,000 images served per second. They'll be moving to a 250,000 IOPS RamSan to handle the load. Also upgraded to a core database machine with 512 GB of RAM, 32 CPU’s, SQLServer 2008 and Windows 2008.
Update 2: This seems to be a POF Peer1 love fest infomercial. It's pretty content free, but the production values are high. Lots of quirky sounds and fish swimming on the screen.
Update: by Facebook standards Read/WriteWeb says POF is worth a cool one billion dollars. It helps to talk like Dr. Evil when saying it out loud.

PlentyOfFish is a hugely popular on-line dating system slammed by over 45 million visitors a month and 30+ million hits a day (500 - 600 pages per second). But that's not the most interesting part of the story. All this is handled by one person, using a handful of servers, working a few hours a day, while making $6 million a year from Google ads. Jealous? I know I am. How are all these love connections made using so few resources?

Site: http://www.plentyoffish.com/

Information Sources

- Channel9 Interview with Markus Frind
- Blog of Markus Frind
- Plentyoffish: 1-Man Company May Be Worth $1Billion

The Platform

- Microsoft Windows
- ASP.NET
- IIS
- Akamai CDN
- Foundry ServerIron Load Balancer

The Stats

- PlentyOfFish (POF) gets 1.2 billion page views/month, and 500,000 average unique logins per day. The peak season is January, when it will grow 30 percent.
- POF has one single employee: the founder and CEO Markus Frind.
- Makes up to $10 million a year on Google ads working only two hours a day.
- 30+ Million Hits a Day (500 - 600 pages per second).
- 1.1 billion page views and 45 million visitors a month.
- Has 5-10 times the click through rate of Facebook.
- A top 30 site in the US based on Competes Attention metric, top 10 in Canada and top 30 in the UK.
- 2 load balanced web servers with 2 Quad Core Intel Xeon X5355 @ 2.66Ghz), 8 Gigs of RAM (using about 800 MBs), 2 hard drives, runs Windows x64 Server 2003.
- 3 DB servers. No data on their configuration.
- Approaching 64,000 simultaneous connections and 2 million page views per hour.

- Internet connection is a 1Gbps line of which 200Mbps is used.
- 1 TB/day serving 171 million images through Akamai.
- 6TB storage array to handle millions of full sized images being uploaded every month to the site.

What's Inside

- Revenue model has been to use Google ads. Match.com, in comparison, generates $300 million a year, primarily from subscriptions. POF's revenue model is about to change so it can capture more revenue from all those users. The plan is to hire more employees, hire sales people, and sell ads directly instead of relying solely on AdSense.
- With 30 million page views a day you can make good money on advertising, even a 5 - 10 cents a CPM.
- Akamai is used to serve 100 million plus image requests a day. If you have 8 images and each takes 100 msecs you are talking a second load just for the images. So distributing the images makes sense.
- 10’s of millions of image requests are served directly from their servers, but the majority of these images are less than 2KB and are mostly cached in RAM.
- Everything is dynamic. Nothing is static.
- All outbound Data is Gzipped at a cost of only 30% CPU usage. This implies a lot of processing power on those servers, but it really cuts bandwidth usage.
- No caching functionality in ASP.NET is used. It is not used because as soon as the data is put in the cache it's already expired.
- No built in components from ASP are used. Everything is written from scratch. Nothing is more complex than a simple if then and for loops. Keep it simple.
- Load balancing

- IIS arbitrarily limits the total connections to 64,000 so a load balancer was added to handle the large number of simultaneous connections.

Adding a second IP address and then using a round robin DNS was considered, but the load balancer was considered more redundant and allowed easier swap in of more web servers. And using ServerIron allowed advanced functionality like bot blocking and load balancing based on passed on cookies, session data, and IP data.
- The Windows Network Load Balancing (NLB) feature was not used because it doesn't do sticky sessions. A way around this would be to store session state in a database or in a shared file system.
- 8-12 NLB servers can be put in a farm and there can be an unlimited number of farms. A DNS round-robin scheme can be used between farms. Such an architecture has been used to enable 70 front end web servers to support over 300,000 concurrent users.
- NLB has an affinity option so a user always maps to a certain server, thus no external storage is used for session state and if the server fails the user loses their state and must relogin. If this state includes a shopping cart or other important data, this solution may be poor, but for a dating site it seems reasonable.
- It was thought that the cost of storing and fetching session data in software was too expensive. Hardware load balancing is simpler. Just map users to specific servers and if a server fails have the user log in again.
- The cost of a ServerIron was cheaper and simpler than using NLB. Many major sites use them for TCP connection pooling, automated bot detection, etc. ServerIron can do a lot more than load balancing and these features are attractive for the cost.

- Has a big problem picking an ad server. Ad server firms want several hundred thousand a year plus they want multi-year contracts.
- In the process of getting rid of ASP.NET repeaters and instead uses the append string thing or response.write. If you are doing over a million page views a day just write out the code to spit it out to the screen.
- Most of the build out costs went towards a SAN. Redundancy at any cost.
- Growth was through word of mouth. Went nuts in Canada, spread to

UK, Australia, and then to the US.

- Database

- One database is the main database.
- Two databases are for search. Load balanced between search servers based on the type of search performed.
- Monitors performance using task manager. When spikes show up he investigates. Problems were usually blocking in the database. It's always database issues. Rarely any problems in .net. Because POF doesn't use the .net library it's relatively easy to track down performance problems. When you are using many layers of frameworks finding out where problems are hiding is frustrating and hard.
- If you call the database 20 times per page view you are screwed no matter what you do.
- Separate database reads from writes. If you don't have a lot of RAM and you do reads and writes you get paging involved which can hang your system for seconds.
- Try and make a read only database if you can.
- Denormalize data. If you have to fetch stuff from 20 different tables try and make one table that is just used for reading.
- One day it will work, but when your database doubles in size it won't work anymore.
- If you only do one thing in a system it will do it really really well. Just do writes and that's good. Just do reads and that's good. Mix them up and it messes things up. You run into locking and blocking issues.
- If you are maxing the CPU you've either done something wrong or it's really really optimized. If you can fit the database in RAM do it.

- The development process is: come up with an idea. Throw it up within 24 hours. It kind of half works. See what user response is by looking at what they actually do on the site. Do messages per user increase? Do session times increase? If people don't like it then take it down.
- System failures are rare and short lived. Biggest issues are DNS issues where some ISP says POF doesn't exist anymore. But because the site is

free, people accept a little down time. People often don't notice sites down because they think it's their problem.

- Going from one million to 12 million users was a big jump. He could scale to 60 million users with two web servers.
- Will often look at competitors for ideas for new features.
- Will consider something like S3 when it becomes geographically load balanced.

Lessons Learned

- You don't need millions in funding, a sprawling infrastructure, and a building full of employees to create a world class website that handles a torrent of users while making good money. All you need is an idea that appeals to a lot of people, a site that takes off by word of mouth, and the experience and vision to build a site without falling into the typical traps of the trade. That's all you need :-)
- Necessity is the mother of all change.
- When you grow quickly, but not too quickly you have a chance grow, modify, and adapt.
- RAM solves all problems. After that it's just growing using bigger machines.
- When starting out keep everything as simple as possible. Nearly everyone gives this same advice and Markus makes a noticeable point of saying everything he does is just obvious common sense. But clearly what is simple isn't merely common sense. Creating simple things is the result of years of practical experience.
- Keep database access fast and you have no issues.
- A big reason POF can get away with so few people and so little equipment is they use a CDN for serving large heavily used content. Using a CDN may be the secret sauce in a lot of large websites. Markus thinks there isn't a single site in the top 100 that doesn’t use a CDN. Without a CDN he thinks load time in Australia would go to 3 or 4

seconds because of all the images.

- Advertising on Facebook yielded poor results. With 2000 clicks only 1 signed up. With a CTR of 0.04% Facebook gets 0.4 clicks per 1000 ad impressions, or .4 clicks per CPM. At 5 cent/CPM = 12.5 cents a click, 50 cent/CPM = $1.25 a click. $1.00/CPM = $2.50 a click. $15.00/CPM = $37.50 a click.
- It's easy to sell a few million page views at high CPM’s. It's a LOT harder to sell billions of page views at high CPM’s, as shown by Myspace and Facebook.
- The ad-supported model limits your revenues. You have to go to a paid model to grow larger. To generate 100 million a year as a free site is virtually impossible as you need too big a market.
- Growing page views via Facebook for a dating site won't work. Having a visitor on you site is much more profitable. Most of Facebook's page views are outside the US and you have to split 5 cent CPM’s with Facebook.
- Co-req is a potential large source of income. This is where you offer in your site's sign up to send the user more information about mortgages are some other product.
- You can't always listen to user responses. Some users will always love new features and others will hate it. Only a fraction will complain. Instead, look at what features people are actually using by watching your site.

Related Articles

- MySpace also uses Windows to run their site.
- Markus Frind's posts on Webmaster World.
- And the Money Comes Rolling In by Max Chafkin
- How I started A Dating Empire by Markus Frind

Thanks to Erik Osterman for recommending profiling PlentyOfFish.