Fun and Games With Googlebot

Posted At : December 14, 2011 4:56 PM | Posted By : Mark Kruger
Related Categories: ColdFusion, Hosting and Networking

When planning for scalability one of the things that is sometimes left out is the impact of indexing bots on your site. If you have a news or ecommerce site that is constantly changing, you definitely *want* bots to be indexing your site. How else are the latest and greatest products or stories going to show up in organic searches after all? But you also want bots to be well behaved. It would great if you could greet the bots at the door and say "Hey... it's 2:00am, not much going on so index to your heart's content." Or, "Whoa there fella - do you have a reservation? This is Cyber Monday and I'm afraid all our seats are full for paying customers. Can you come back in 12 hours?" But that sort of smart interaction is sadly not in the cards. Some bots have defined rules, some do not. Some honor things you put in the robots.txt file others do not. So here are some tips that might save you some time.

Google Webmaster Tools

Google has a "webmaster center" you can use to fine tune how they index your site - and believe me *you should use it!* Why? Because in fact, left uncontrolled on a busy site with lots of changes Googlebot will index wildly with as many as 10 requests per second. To get "hooked up" with Google's webmaster tools system you have to sign in **here** with your Google account. Then you add your domains. You have to "prove" you own a domain by placing a file that Google provides (a simple HTML file) in your root directory. Then you can adjust your settings. This **blog post** is a pretty good one for describing how to use the tools.

Setting Crawl Delay

Slowing down Googlebot is step one - and sometimes it's all you need. But you can affect both Slurp (that's what Yahoo calls it's crawler) and MSN by using a special syntax in your robots.txt file.

```
User-agent: Slurp

Crawl-delay: 5



User-agent: msnbot

Crawl-delay: 10
```

This will create a 5 or 10 second delay between crawler requests. This **post** from SEO Book has some additional "robots.txt" tips that are quite useful.

Session Creation

It's not obvious but when considering bots you need to carefully examine your *session creation strategy*. Let's consider for a moment how Googlebot "indexes" your site. There's no browser. There's no user. Instead your site is one of a few hundred million that the Googlebot engine has to get through in a day. A request is made for the home page, the home page is indexed and then parsed for additional links and then these links are also retrieved, indexed parsed and so on. It's the giant virtual circle of life.

On that first request your system will see a "new" request and create a new session for that request. That's good right? But what happens on subsequent requests? Keep in mind that sessions are most often tied to cookies, but Googlebot is not going to return that jsessionID or cfid/cftoken or custom uuid cookie you tried to set. Instead it's going to stick to links it finds within your site. Googlebot wants site *content* and does *not* want any specific personalization or activity that might be tied to a session. The result? Every time Googlebot hits a page a new session is generated. If you are using some form of "roll-your-own" session management (a UUID in a database for example), your database is going to be busily minting new sessions with every single bot requests unless you take steps to mitigate the situation.

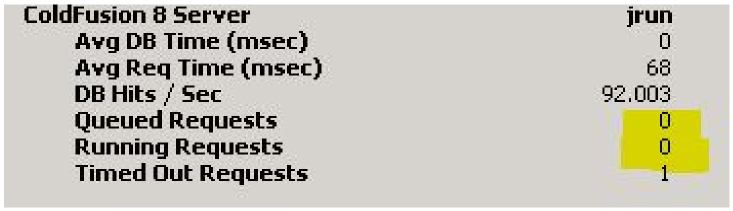
Why Can't I Detect it?

Ok... so why is it difficult to see excessive bot traffic? Can't I just pour through the logs and find a bunch of simultaneous entries from a particular user agent or IP address? If you have a *low* traffic site the answer is yes, probably. But if your site is high traffic the answer might be no, not likely. Why? I'm glad you asked. It boils down to two seemingly contradictory facts.

1. On a busy site the amount of bot traffic in the logs can be small on a relative basis.
2. Bot traffic is exponentially more expensive than other traffic because it taps your application specifically.

I Esplain It To You Lucy

Wait a minute Muse! You tell us bots are obscured in the logs and hard to detect, but they also have some super power and their requests are more expensive. What gives? To illustrate, let me tell you a story. I was recently troubleshooting a couple of familiar servers. They were running ColdFusion 8 enterprise - well provisioned, large heap, plenty of RAM and fast disks. The servers were acting strangely. The number of CF requests were pegging and queuing constantly. In case you don't know what that means, consider these perfmon counters I like to watch. It's the group of perfmon counters that ships with the ColdFusion 8 windows version.

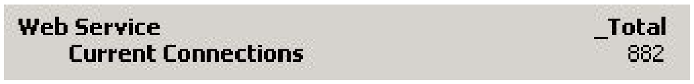


ColdFusion 8 Server	jrun
Avg DB Time (msec)	0
Avg Req Time (msec)	68
DB Hits / Sec	92.003
Queued Requests	0
Running Requests	0
Timed Out Requests	1

The 2 highlighted counters - running requests and queued requests - are like the canary in the coal mine. Running requests climbing steadily is almost never a good thing. When running requests reaches the top of the *Simultaneous Requests* (in the CF Admin under request settings) CF stops trying to process them concurrently and moves new requests to a "queue". At that point the *Queued Requests* will begin to climb. If you want to try to see how it works, do a load test on your dev site and stop your database server in the middle of the test. You will see exactly what I mean.

Indeed, the database is often the first place I look when this specific problem comes up (requests climbing and queuing). If the DB is in trouble, bogging down, in need of indexing etc. - this is the behavior I often see. Looking at the DB I discovered 2 things. First, the transaction log file for the main 8gig database file (MSSQL) was 300+ gigabytes. That's a hefty log file to be sure. Typically a log file of this size indicates 2 possibilities. Either the site is receiving an excessive number of updates, inserts and deletes (a lot of "activity" being logged), or the full backups haven't been run correctly in a while. Running a successful "full" backup will clear out the TLog space and keep it from growing excessively. Checking the backups I found them in good order with appropriate log and verification entries. So the DB was actually doing a whole lot of work.

It was a busy ecommerce site and it was busy to be sure - but I had yet to understand *what kind* of traffic I was seeing. To my eye the traffic was still well within our load tested baseline. To see a "real time" glimpse of traffic I like to turn again to Windows Performance Monitor. Under "Web Service" there is a counter called "current connections". It shows the current "yet to expire" number of HTTP connections hitting the web server at a given time. These 2 servers tested out to be able to handle around 2000 connections under load (give or take a few hundred) and they were currently running at about 800+ connections per second.



```
Web Service                          _Total
      Current Connections              882
```

During times when requests would queue I was *not* seeing what I considered excessive connection traffic - which is why I thought perhaps it was networking or the DB Server. Finally I had the notion to take a look at the session table on this application. The app used a "roll your own" approach for sessions through - a robust solution through a load balancer, but one that can definitely pressure the database. Now the IP address was stored with the session so the first thing I noticed was a whole lot of "new" sessions being created against the same IP address or addresses. In fact, one IP address minted 5000 sessions in less than 2 and a half minutes. That is some aggressive requesting. That IP address (which happened to be Googbot's IP), was creating a session with each request - sending insert and update statements to our DB willy-nilly, slowing down the DB server (due to locking) as causing an exponential impact on our system. I excluded that IP from being allowed "in" to the application and Voila! Things improved immediately. But why could I not "see" that this was a bot problem right out of the gate?

The Muse Forgets, Then Remembers

In a huge head-slapping moment I recalled how indexing works again. Googlebot is not a browser remember. A browser selects a page, parses the pages, then generates more requests for all the resources the page needs to look correct - like CSS, JS, images, etc. But an indexer hit's a page (a CFM page remember?) then finds links (to more CFM pages) and hits them as well. But unlike a browser it doesn't go out and find all the rest of these resources and bring them in. It's requests are lean and mean and to the point - only hitting my application pages. A regular user might account for 50, 80 even 100 http requests when they hit a single page. That's especially true in ecommerce where many pages have dozens of images for products. But a bot hitting the *same* page generates only a single request. The request is just as expensive *on the application* but it does not generate a corresponding bump in *connections* to the web server - at any rate not in a way that is easily visible. The higher trafficked the site, the more likely it is that your bot traffic is buried (both in real time and the logs) in a plethora of requests for "everything else" that makes your site the lovely and engaging and helpful oasis on the web that we all love.

Conclusion

We are currently devising a bot sniffing strategy that ties a bot to a dummy session so it has zero impact on the database. Meanwhile, I'm moving bots up on my priority list in my troubleshooting. In this case a single bot (from Google dang it! I love Google!) was wreaking havoc on the system. From now on we check that first when I see this behavior. And now I'm sure my savvy readers will all want to chime in and give me their take on the topic. I look forward to learning from them. Just be nice. :)