ARE SPIDERS EATING YOUR SERVERS?

THE IMPACT OF THEIR UNEXPECTED LOAD AND HOW TO COUNTER IT

Charlie Arehart, Independent Consultant
CF Server Troubleshooter
charlie@carehart.org
@carehart (Tw, Fb, Li, Slack)

Updated July 17, 2017

SOME INTRO QUESTIONS FOR YOU

THERE IS GOOD NEWS

- Good news: there are solutions to mitigate impact, perhaps reduce load
- That said, some automated requests are getting smarter, harder to control
- Beware: think your intranet/private/login-required site is safe from impact?
- We'll cover all this and more in this talk

ABOUT ME

- Focus on CF server troubleshooting, as an independent consultant
  - Satisfaction guaranteed. More on rates, approach, etc at carehart.org/consulting
- Love to share info, with my clients and the community
  - Contributor to/creator of many CF community resources
    - Online CFMeetup, CF411.com, UGTV, CF911.com, CFUpdate.com, and more
- I'm also manning the Intergral (FusionReactor) booth for them

TOPICS

- Understanding automated requests
  - The nature of such automated requests (many, varied, not always friendly)
  - How we can generally identify such requests
  - Their generally unexpected volume
- The impact of such request volume, CF-specific and more generally
- Observing the volume in your environment
- Dealing with automated requests: tools and techniques
  - Preventing undesirable ones
  - Mitigating the impact of expected ones, CF-specifically and more generally
- Resources for more
- Slides at carehart.org/presentations

UNDERSTANDING AUTOMATED REQUESTS

THE NATURE OF SUCH AUTOMATED REQUESTS: CRAWLERS

- Of course most common automated agents are search engine crawlers
  - The intent/approach of such search engine crawlers/bots/spiders
- There are many:
  - Some legit and desirable (google, bing, yahoo, etc.)
  - Some legit but maybe not your market: Yandex (Russian search engine), Baidu (China, also SoGou, Youdau), Goo (Japan), Naver (Korea), etc.
  - Some may be legit but perhaps unfamiliar to you (Rogerbot, for seomoz.org, mj12bot, for majestic12.co.uk)
- Analogy: restaurant scrambling to serve crush of non-paying reviewers
- …

THE NATURE OF SUCH AUTOMATED REQUESTS: CRAWLERS (CONT.)

- Some crawlers visit your site for other purposes:
  - Some are looking to find **copyright violations** (maybe ok)
  - Some grab ecommerce site prices **to show elsewhere** (may be dubious)
  - Some grab content to **sell to competitors** context about your site/business (not cool)
- Then there are **RSS/atom readers**/services, calling into feeds on your sever
- And you may **expose APIs, web and REST services** that are called in auto. ways

- And before you feel safe with non-public/intranet site, behind firewall or login
  - Beware: site may be crawled by **internal search appliances**
- But that's not all (that can affect both intranet and traditional web sites)…

THE NATURE OF SUCH AUTOMATED REQUESTS: OTHER CHECKS

- And how about **load balancer health checks**?
- And **monitoring checks** (setup by you, your IT folks, or your clients)?
- Consider also **site security scans**
  - May be run by folks in your IT org, to find vulnerabilities
  - These often run requests at high rates, trying many ways to "break in"
- Analogy: restaurant scrambling to serve free-loading family members

THE NATURE OF SUCH AUTOMATED REQUESTS: ERRORS

- And consider also the added impact of error handling of those, or 404s
- Still another cause: **coding mistakes** leading to repeated requests
  - Such as a runaway ajax client call

THE NATURE OF SUCH AUTOMATED REQUESTS: MISCREANTS

- And of course **hackers, thieves, miscreants** attempting increasing harm:
  - Comment and other forms of spam
  - Theft of content
  - Break-in/takeover of accounts
    - Including outsiders running security scans to find vulnerabilities
  - Fraudulent transactions
  - Denial of service (ddos)
    - Which could be as simple as them running load test tools against your server
- Analogy: restaurant scrambling to serve folks stealing from the register, blocking the door, etc.

- OK, so now we know some common kinds of automated requests…

IDENTIFYING SUCH BOTS

- Requests typically self-identify with a **"user agent" header**
  - Browsers identify the kind of browser they are (Chrome, FF, Safari, Opera, IE, etc.)
  - And most legit bots will also provide a user agent (UA) string
- Some bots also provide a URL in the UA as well
  - A page to explain perhaps what they do, how to manage their requests
- Nice free web site to lookup and better understand UA strings
  - https://www.distilnetworks.com/bot-directory/
  - Gives ratings (good/bad), known IP ranges, more

IDENTIFYING SUCH BOTS (CONT.)

- Do beware: a requestor can **lie about their user agent**
  - Some may look like "real browser", others like "legit spider", to throw you off
  - If you see a "Googlebot" UA from an IP on Amazon, they're a liar!
- Still others may provide no user agent at all
  - And we could use that against them, in **rejecting requests without any UA**
- Let's talk about other ways to identify them, then how we may handle them

BOT CHARACTERISTICS WE MIGHT WATCH FOR TO BLOCK THEM

- Most automated agents also present **no cookie** (important impact, later)
  - Of course, a real first-time user will also have no cookie from your site
  - But if we get many frequent requests from same IP with no cookie, we might count that against them
- Many automated requests might show **no "referrer" header**
  - Of course, neither will a request where someone types your URL into a browser
- IP addresses of many requests at once may be same, or in a small range
  - Or **may have same UA but totally random IPs**, which could be suspicious
- We'll revisit consideration of such characteristics under "mitigation" later

- So again, why might all this be a problem?...
- Most of these automated requests (of all types) tend to come every day
  - Generally hitting ALL your site pages
  - And a given single "page" may be reached by different URLs (bot won't know)
- Not unusual for folks to have "paging" links, accessing all pages of a type
  - For instance, all products, and as viewed over all categories, then all vendors, etc
- And remember, each kind of bot may visit thousands of your pages per day
- This is why it's not unusual to find these being 80% of site requests!

- And so what? ....

THEIR GENERALLY UNEXPECTED VOLUME

THE IMPACT OF SUCH REQUESTS

- Of course, such high volumes of requests have impact on:
  - General compute resources (cpu, memory, disk)
  - Some may be tempted to increase hardware to "handle the site's load"
- Consider also the bandwidth used to serve each page requested
  - And all associated files (CSS, JS, image files)
  - Perhaps millions per day, per bot, day after day ad infinitum
  - Someone's paying for that bandwidth!
- Then consider impact on entire infrastructure
  - Web server, application server, database server, san/nas, network, perhaps mail server, etc.
- For CFML pages specifically, impact is even more significant…

GENERAL IMPACT

CF-SPECIFIC IMPACT: SESSIONS

- First, **session and client creation**
  - Talking here about CF sessions (or J2EE sessions), stored in memory of CF/heap
  - Not referring to "web sessions" as tracked by web servers, Google Analytics, etc
- CF sessions are used to track data for a user across many requests
  - Based on sessionid cookie being passed from client on each request
- But most automated agents **send no cookie**, thus creating a new session/client for EACH page requested!
  - Not unusual for me to help folks find 20k, 100k, or more "active" sessions!

CF-SPECIFIC IMPACT: SESSIONS (CONT.)

- Such high session count could have **impact on heap use** within CF, of course
  - And "weight" of session influenced by what your code puts into session
- Consider also **session timeout**: how long unused sessions remain in memory
  - May be hours or even days in some setups
- Max and default timeout set in CF admin, of course
  - Can be overridden in application.cfc/cfm
- Longer timeout X more mem per session X more sessions = more heap

CF-SPECIFIC IMPACT: SESSIONS (CONT.)

- Still worse: consider your session startup code, running for each new "session"
  - Talking about onsessionstart in application.cfc
    - Or perhaps code in application.cfm within a test for session existence
  - You may create queries, CFCs, arrays/structs, stored in session scope for user
- Consider then the incredibly high rate of executions per minute, hour, day
  - May be executed FAR more often than the developer ever anticipated

CF-SPECIFIC IMPACT: CLIENT VARS

- Consider also impact if your code enables client variables (clientmanagement="yes")
  - Default behavior is that each request creates/updates client repository "global variables" (hitcount, last visit)
  - So that's still more activity per request
- Worse: such automated requests create NEW client repo entries on EACH request!
  - Bad enough if these are stored in a database: lots of i/o, possible congestion
  - Again to track information for what may be just a single visit ever
- Worst still if client vars might be stored in registry
  - Or worst of all, if on *nix where such "registry" processing is really just a "reg" file!

CF-SPECIFIC IMPACT: ERRORS & MORE

- Consider also impact of spiders/bots on your 404 and error handing
  - Automated agents may call many pages that don't exist (repeatedly)
  - Or they may call pages in an unexpected "order", triggering errors
  - Or their high volume may create still more errors
- Consider needless filling of caches (query cache, template cache, etc)
- Consider also impact on cfhttp calls your code may make to other sites
  - Maybe to obtain information, or to share it, on each/many/most requests
  - Such high volume of automated requests may cause YOU to be abusing others
    - Your requests may be throttled by such other sites, affecting your "real" users

CF-SPECIFIC IMPACT (CONT.)

- So I hope I've made the case that you may well need to worry
  - How can you know if you should?

OBSERVING VOLUME IN YOUR ENVIRONMENT

OVERVIEW OF A COUPLE OF SIMPLE WAYS

- There are a couple of relatively straightforward ways to observe such traffic
- You may know that some built-in tools log every request
  - And tools exist (free and commercial) to help analyze such logs
  - Such logs can also be configured to track user agent, cookies, referrer
- Some tools also let you track count of sessions
- Let's look at these a bit more closely

ANALYZING LOGGING OF REQUESTS

- Web server logs (IIS, Apache, nginx) track every request
  - Of course, they track requests of every type: images, js, css, etc.
  - These can optionally be configured to track user agent, cookies, referrer
- Tools exist to monitor such web server logs, track web site "traffic"
  - Some are more "marketing" oriented, may literally hide spider/bot traffic!
  - Some may well distinguish spider traffic
- Other tools can analyze an CSV logs, which is useful because …

ANALYZING LOGGING OF REQUESTS (CONT.)

- ColdFusion (Tomcat) "access" logs can also be enabled to track CF requests
  - Turned on by default in CF10, off by default in CF11, 2016
  - These track ONLY CF page requests, of course, assuming CF is behind a web server
  - These can also be configured to track user agent, cookie, referrer
- FusionReactor logs also track every request
  - And can be configured to track UA; already tracks incoming session cookies if any
- Tools for log analysis: http://www.cf411.com/loganal

TRACKING OF REQUESTS VIA BEACONS

- Again there are tools/services that can track visits via tracking beacons
  - You implement a small bit of javascript in your code
  - When that page is visited, a request is made from the client to some server service, which tracks requests
  - Examples: Google Analytics, Google and Bing Webmaster Tools, and more
- And better versions of such tools do distinguish spider/bot traffic
- Do beware, some "clients" won't execute the Javascript that triggers such tracking
  - And so some such automated requests may not be tracked at all

TRACKING SESSIONS AND MORE

- CF10 and above track session count in metrics.log; enabled in CF Admin
- FusionReactor and CF Enterprise Svr Monitor track current count of CF sessions
  - FR also tracks session count over time and across restarts, in realtimestats.log
- Beyond sessions, CF tracks cfhttp calls in cfhttp.log
- 404s and application errors tracked in application.log, or handled by your app

- So once you confirm you DO have lots of automated traffic, how do you handle it?...

DEALING WITH AUTOMATED REQUESTS: TOOLS AND TECHNIQUES

PREVENTING UNDESIRABLE ONES

- First thought may be "block" undesirable requests by IP address
  - Beware: most come from a block of them (and bad guys may falsify IP)
  - Becomes game of "whack-a-mole"
- May think to block by user agent
  - Beware: some bad guys present legit-looking user agents
- The black hats are trying always to stay a step ahead of the white hats
  - Consider also Perimeterx's "4 generations of bots"
    - https://www.perimeterx.com/resources/4th-gen-bots-whitepaper
- Still, for a large amount of most common automated traffic, these simplistic approaches may be better than doing nothing (more in a moment)

MITIGATING THE IMPACT OF EXPECTED ONES, MORE GENERALLY

- Simplistic solutions to manage such agents may exist already in your env
  - Robots.txt: simple, but could be ignored
  - Web server IP blocking features: like playing whack-a-mole
  - URL rewrite tools could block requests by a variety of characteristics
  - IIS request filtering can block by user agent string
- Any of these might work just fine for some, but may be too simplistic for many
- There are still other options…

MITIGATING THE IMPACT OF EXPECTED ONES, MORE GENERALLY (CONT.)

- Some firewalls (software or hardware) can manage bots
  - Some web app firewall solutions in or available for most web servers can help
- Indeed, some cloud services offer protections against spiders/bots/hacks
  - https://aws.amazon.com/blogs/aws/new-aws-waf/
  - https://azure.microsoft.com/en-us/blog/azure-web-application-firewall-waf-generally-available/
- You could also consider also web content caching proxy solutions
  - To at least reduce impact reaching your server
- Or we can get still more sophisticated about this specific problem…

- There are tools/services that detect/mitigate negative bot impact
  - Some free, some commercial
    - Some easily implemented, others even offered as SAAS with virtually no change
  - Examples: Distil, Incapsula, Shieldsquare, PerimeterX, Akamai
- These companies are making it their job to watch for and block bots
  - Even the most sophisticated ones
  - Most offer options to report-only at first, and then tweak/turn on to block bad guys
- And may want to consider those focused more on blocking hacks rather than bots, per se
  - Shape Security, Securi, Cloudflare, etc

- Now on to more CF-specific mitigations…

MITIGATING THE IMPACT OF EXPECTED ONES, MORE GENERALLY (CONT.)

MITIGATING THE IMPACT OF EXPECTED ONES, CF-SPECIFICALLY

- May want to modify session timeout on per-request basis, lower for bots
  - Consider watching programmatically for characteristics like:
    - No user agent, no referrer, and no cookie
- Can implement either in:
  - In application.cfm, where you can vary cfapplication sessiontimeout
  - In application.cfc, may not want to vary this.sessiontimeout (applies to loaded app)
    - Instead, could handle in onrequestend
    - Can either "invalidate" session, or lower session timeout for that request only via java

MITIGATING THE IMPACT OF EXPECTED ONES, CF-SPECIFICALLY (CONT.)

- May also want to reconsider coding choices in your session startup code
  - Maybe don't store large amounts of info at session startup (queries, arrays, structs, CFCs) if request is determined to be for an automated agent
  - Given that session won't be re-used anyway by most automated request agents
- Also, reconsider error handling, to only respond to cfm/cfc pages
  - And maybe html pages if you must, but not image/css/js files
- Consider admin config options related to sessions and/or client variables
  - Session timeout: reconsider default/max times, and times set per app
  - Reconsider client var storage options (cookie vs db/registry)

MITIGATING THE IMPACT OF EXPECTED ONES, CF-SPECIFICALLY (CONT.)

- Could also add code to at least throttle excessively frequent requests
  - http://www.carehart.org/blog/client/index.cfm/2010/5/21/throttling_by_ip_address
  - Note that ContentBox incorporates a variant this code, enabled by a checkbox
- "Outside the box" possibility (for CF Enterprise, Lucee/Railo)
  - Create a separate instance to JUST serve automated traffic
  - Direct such traffic there with web server rewrite features

- So, phew, that's a lot to take in!
  - Understanding issue, mitigating it
- I've provided a broad overview
  - You may want to dig in to the topic further
  - There are many resources that focus on the topic generically in significant depth

- http://www.itproportal.com/2015/04/25/7-ways-bots-hurt-website/
- https://searchenginewatch.com/sew/news/2067357/bye-bye-crawler-blocking-parasites
- https://blog.cloudflare.com/introducing-scrapeshield-discover-defend-dete/
- https://www.digitalcommerce360.com/2016/11/11/bad-bots-are-real-heres-how-hayneedle-fought-them/
- https://www.incapsula.com/blog/bot-traffic-report-2016.html
- http://scraping.pro
- https://resources.distilnetworks.com/
- https://www.incapsula.com/resources/
- https://www.perimeterx.com/resources/
- https://www.cloudflare.com/resources/

RESOURCES

SUMMARY

- The nature, volume and impact of automated requests is often hidden
  - It is possible to observe the volume, mitigate the impact, perhaps easily
  - Can lead to a substantial improvement in performance, bandwidth savings
- Again, my contact info for follow-up:
  - Charlie Arehart
  - charlie@carehart.org
  - @carehart (Tw, Fb, Li, Slack)
  - carehart.org/consulting
- Thanks, and hope you've enjoyed the rest of the conference
  - Come see me at the FusionReactor booth, where I am manning it for them