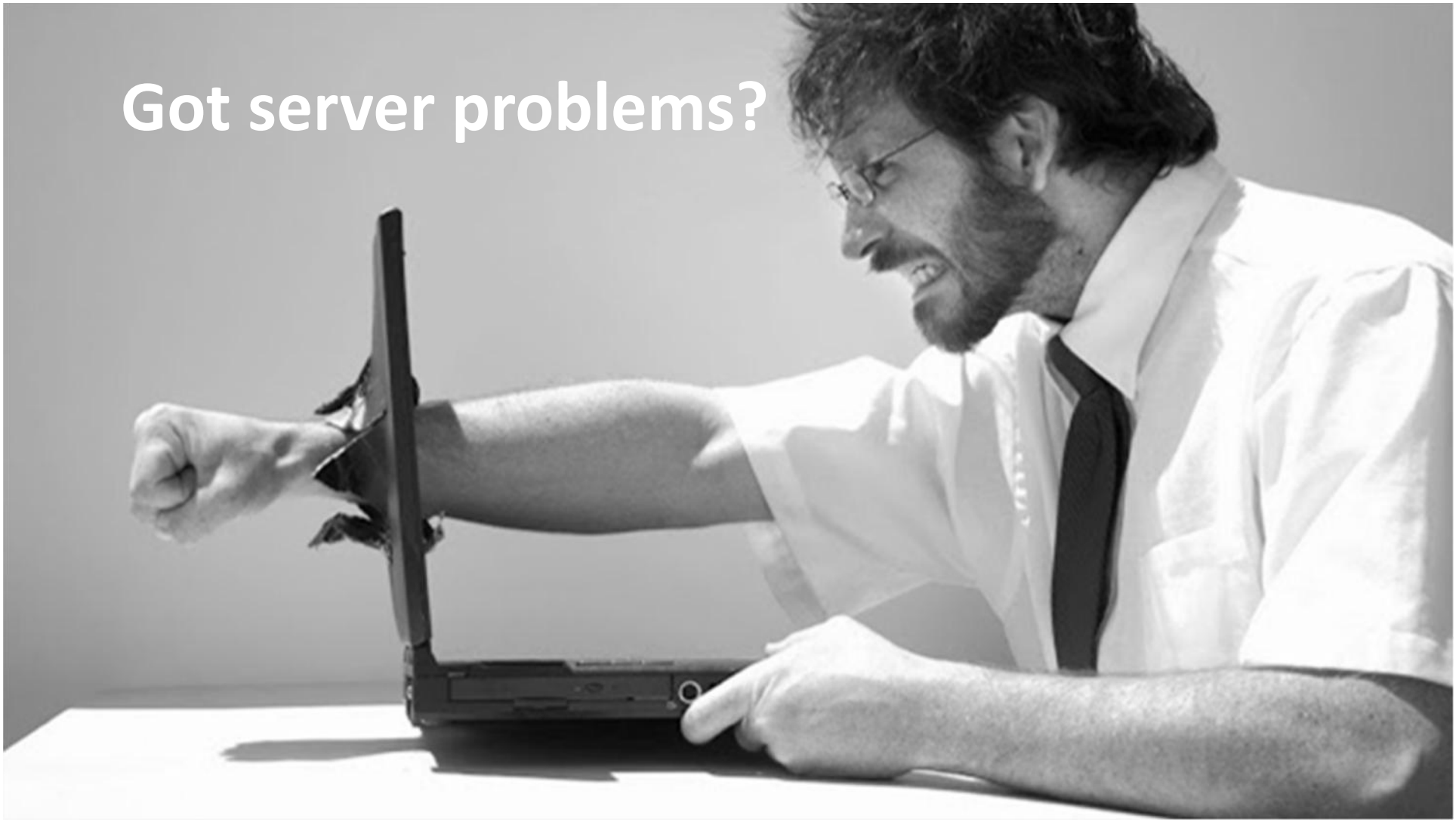
Are spiders eating your servers?

The impact of their unexpected load and how to counter it

Charlie Arehart, Independent Consultant
Server Troubleshooter

Updated May 2, 2017

SSWUG.org
Your Information Technology Answers Are Here


Got server problems?

There's 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 intranet/private/login-required site is safe from impact?
- We'll cover all this and more in this talk
- To be clear: this discussion is not really specific to SQL Server
  - More about impact of these things that would FLOW to SQL Server (or any DB) and from any web/application server
- This is a talk to get you thinking, point you in perhaps new directions

Topics

- Understanding automated web 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, app/DB server-specific & more generally
- Observing the volume in your environment
- Dealing with automated requests: tools and techniques
  - Preventing undesirable ones
  - Mitigating the impact of expected ones
- Resources for more
- Slides at carehart.org/presentations

UNDERSTANDING AUTOMATED REQUESTS

SSWUG.org
Your Information Technology Answers Are Here

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

- 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, 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 to handle them

Bot characteristics we might watch for to block them

- Most automated agents also present **no cookie** (important impact, later)
  - Of course, 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 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

Their generally unexpected volume

- 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? ….

THE IMPACT OF SUCH REQUESTS

General Impact

- Of course, such high volumes of requests have impact on:
  - General compute resources (cpu, memory, disk), on app server or SQL Server
  - 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, network, san/nas, perhaps mail server, etc.

- For web app server pages specifically, impact is even more significant…

Application server-specific impact: sessions

- First, **session creation**
  - Talking here about app server sessions, stored in memory of app server
  - Not referring to “web sessions” tracked by web servers, Google Analytics, etc
- App server 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 for EACH page requested!
  - Not unusual for me to help folks find 20k, 100k, or more “active” sessions!

App server-specific impact: sessions (cont.)

- Such high session count could have **impact on app server memory use**
  - And “weight” of session influenced by what your code puts into session
- Consider also **session timeout**: how long sessions remain in memory
  - May be hours or even days in some setups
- Longer timeout X more mem per session X more sessions = more memory

App server-specific impact: sessions (cont.)

- Still worse: consider session startup code, running for each new “session”
  - May create queries, objects, 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

App server-specific impact: errors and 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 just their high volume may create still more errors
- Consider needless filling of caches, or SQL Server buffer pools
- Consider also impact on httpclient 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

App server-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

SSWUG.org
Your Information Technology Answers Are Here

Overview of a couple of simple ways

- There are a couple 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/services track visits via tracking beacons
- 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
- Tools for log analysis: http://carehart.org/cf411/#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

- ASP.NET sessions can be tracked via its Perfmon counters
  - Or if you use any various state server solutions, they offer counts as well

- 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 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 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://azure.microsoft.com/en-us/blog/azure-web-application-firewall-waf-generally-available/
  - https://aws.amazon.com/blogs/aws/new-aws-waf/
- 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…

Mitigating impact of expected ones, more generally (cont.)

- 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 report-only option, can 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 app server-specific mitigations…

Mitigating the impact of expected ones, app server-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
  - Modify httpsessionstate.timeout
- May also want to reconsider coding choices in your session startup code
  - Maybe don't store large amounts of info at session startup (queries, objects, arrays, structs) if request is determined to be for an automated agent
  - Given that session won't be re-used anyway by automated request agents
- Could also add code to throttle excessively frequent requests from an IP

Mitigating the impact of expected ones, app server-specifically (cont.)

- "Outside the box" possibility
  - Create a separate site/server to JUST serve automated traffic
  - Direct such traffic there with web server rewrite features

We're about done…

- 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 focused on the topic generically in significant depth

Resources

- 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/

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
- My contact info for follow-up:
  - @carehart (Tw, Fb, Li, Slack)
  - carehart.org/consulting

- Thanks, and enjoy the rest of the conference
  - Including my other talk, “SQL Server 2016 SP1 Changes the Game”