ACME Corporation

Anvils • Explosives • Rockets • Roller Skates

Performance Optimization and Tuning Recommendations

(Abridged)

January 25, 2023

Author: QA Consultants Performance Engineering Group

QA Consultants USA, Inc.

5706 E Mockingbird Lane Suite 115-23

Dallas, TX 75206

USA

QA Consultants

For nearly three decades, QA Consultants mission has been to help clients understand the quality of their software applications: Does it work for a single user? Does the software meet operational requirements for scalability? Is it secure?

How may we help you?

https://qaconsultants.com
https://qaconsultants.com/solutions-and-services/performance-engineering/

ACME Corporation, Performance Optimization and Tuning Recommendations (Abridged)

QA Consultants Performance Engineering Group
5706 E Mockingbird Lane, Suite 115023. Dallas, Texas 75206 USA

Chief Performance Officer:	James Pulley	jpulley@qac.com	704-351-0117
Head of US Operations:	Brian Bernknopf	bbernknopf@qac.com	512-699-5386

Document number: ACME110722RoadRunner-A

Permission granted for redistribution

Executive Briefing, ACME CFO & CTO

At the request of executive management, QA Consultants examined the usage of ACME.COM, the sales portal for ACME's complete product line. ACME is a multinational corporation with diverse product lines, ranging from consumer roller skates to industrial items such as Anvils and High Explosives to aerospace items such as rockets. In addition to direct sales to key customers, ACME's website is the source of two-thirds of the company's revenue stream.

The goal of this engagement has been to identify opportunities to improve the efficiency of ACME's website, specifically to reduce the cost of operations and improve revenue via an improved customer experience. A summary of our findings follows:

Cost Reduction

Bots The top 200 user connections to your site generate 24.4% of the system load. No sale can be identified for the examined period associated with the addresses in question. Web application firewall (WAF) rules can remove this load from the site. On a conservative basis, this will lower your cloud resource needs by 20% from your current deployed virtual machine set.

Site Configuration Thirty percent of the load on your core website is associated with requests that should be cached by your site cache or CDN service.

- Teapot Service Status, HTTP 418, @ 24.9% of load
- Missing response, HTTP 404, @ 4.9% of load
- Permanent Redirects, HTTP 301, @ 1% of load

We estimate that new bot firewall rules will address half of the load. The residual amount is estimated at 15% of the origin server system load and will be diverted by new Cache/CDN rules. These rules will also result in faster delivery of web page assets to the client and a faster user experience

Rewards Program - Update Rewards This one request is responsible for 23% of the application load on your system. A request to the rewards program occurs in parallel with the default shopping cart on each page for a credentialed site user. This individual request requires optimization, and calls to this request should be reserved when rewards are incremented or decremented, such as using reward points to pay for an item or updating rewards points after a sale. A reduction of 15% is projected associated with these two changes.

We conservatively estimate a total reduction of load on the ACME website servers of 50%, with an improved user experience related to the delivery of web page assets. This enhanced experience typically impacts conversion rates positively.

Revenue Optimization

Conversion is a complex process, but speed is a critical element. Where two websites are identical, the faster one will keep a customer onsite longer with a higher opportunity to convert. Several factors have been identified related to improved conversion in addition to the delivery of assets above

Session Persistence The current session timeout is set to 24 hours at the request of marketing. This session timeout value is too long, locking system resources that could otherwise be allocated to current converting users. The cost of a new session created for the small number of users returning within 24 hours is small compared with the locked resource costs. Most of the users currently viewed as being "on the site" are waiting to time out.

Default Shopping Cart Fewer than 20% of the active sessions use a shopping cart anytime during the visit. Having a default cart, where every user is given a cart upon first access to the system, forces a relationship between the most distant piece of the site architecture, cart storage, and every user. This too-early allocation of a shopping cart acts like a drag anchor behind a high-speed boat to slow the entire site down. You want the user experience delivered out of cache or CDN until the user adds an item to the cart. Moving to a just-in-time cart will reserve the majority of the resources for users on a revenue path

Perpetual Carts In addition to a default cart, ACME stores a copy of the user cart associated with each user session. This storage, combined with a default cart issued to site users, has resulted in millions of empty, stored carts. This slows cart access for carts with content. There is a related issue of old stored carts that have never been re-accessed or contain items no longer in inventory.

Check Out Steps The Current number of steps to check out is seven. With each step, ACME is losing between 30-50% of users. We recommend reducing the number of checkout steps by one, consolidating bill-to and ship-to addresses on one page instead of two steps, with a default of bill and ship as the same. The current drop-off between these two pages is 30%, according to Google Analytics provided by your marketing team. Because of the downstream drop-off, we estimate a 5% increase in sales from this single change.

We estimate an increase of 10% due to faster system responses, a larger resource pool available for high volume days resulting in more users being served, and a reduction in the number of checkout steps.

James L Pulley, III
Chief Performance Officer, QAC

About QA Consultants

For the past thirty years, we have focused on improving software quality for our clients on both a functional and scalability basis. We have taken lessons learned over the past two decades for common performance anti-patterns and converted those to rules that can be used to evaluate the performance of any web-based application.

Our recommendations focus on two particular areas. The first is client response time. The second is server scalability, which impacts the speed at which files are served to the client under high-load conditions. The recommendations in this document are prioritized by the impact on both focus areas above with the joint goals of improving client responsiveness and server utilization.

If your evaluated application or site is running at a cloud provider, such as Amazon, Microsoft Cloud Azure, IBM Cloud, or other cloud providers that charge by resource access will obtain additional benefits: The reduced load on your hosted servers will result in a lower cost of operations for the application or site so hosted. As your resource requirements drop for a given user load, so will your bill.

How we measure your performance

We take a different approach than many organizations. With over two decades of performance engineering experience, we have defined a set of patterns that appear in your HTTP logs that are indicative of performance issues.

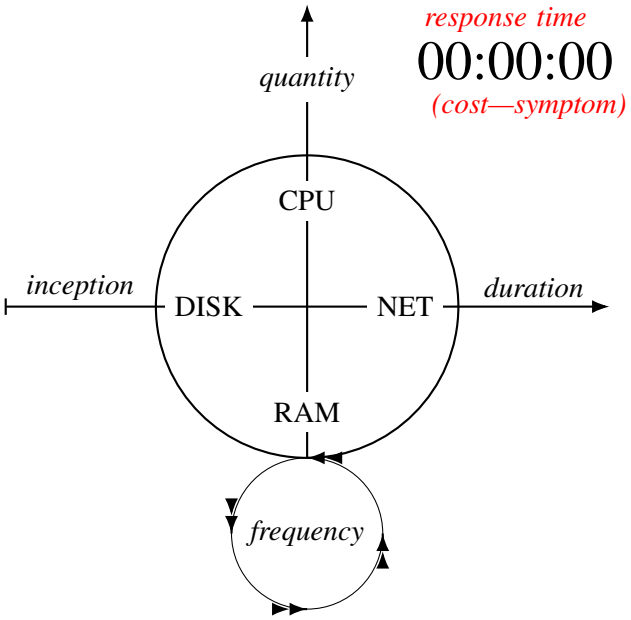


Figure 1: *Standard Performance Engineering Model*

These patterns are related to resource usage for CPU, DISK, MEMORY, & NETWORK. It is how an application makes use of these resources which drives the response time and scalability of a system. When you allocate a resource, how large of a block is allocated, how often you access the resource, and when you release it all affect system performance. We find evidence for use in the logs from your application.

This is not unlike taking your car in for an emissions check, where your car's exhaust is monitored while a technician moves the engine through a defined cycle. The type and number of particulates present in the exhaust indicate issues that need to be addressed for the proper performance of the car.

We are treating your application logs as the exhaust for your system. We then run a set of tests against the log data to uncover the existence of items leading to poor response times for your users and slow responsiveness from your web servers.

We use a custom rules-based analysis engine. Its dashboard capabilities allow us to look at the the output of pattern analysis inquiries against access logs related to cache management, errors, where time is being spent in the system under evaluation, how many users, Bots, etc...

How are we different? We provide actionable intelligence in the report. Where an issue is identified, included is the data on how to fix the problem. This is not just machine learning to identify a pattern or a new pattern but an application of artificial intelligence to help your developers and administrators address the issue.

The following report contains many generic features, and the sections are abbreviated. Wherever an ellipsis ("...") appears in this document, an actual report would have included additional content.

An actual report would contain paths to locations in management consoles, web servers, and queries to databases to resolve the issues.

Why the large margins and the blank space before chapters? Reports rarely include enough white space for collecting observations & thoughts while reading a document. We have included a generous outside margin for your notes. Grab that Sharpie!

Contents

Chapter 1 Impact of Automated Agents & Bots

For the past decade, Imperva has been producing an annual Bot survey of the internet[1], tracking both the percentage of bots relative to human users as well as the ratio of bad bots to good bots based upon behavioral characteristics of the bot. In some cases, such as an eCommerce top 100 example detailed below, the number of bots can equal or surpass human users. This high volume of bots is particularly problematic during times of spot load, where an announcement drives load to a site, for the frequency at which a bot can operate is orders of magnitude faster than a human user. This results in a crowding-out effect for human users of a site/system. From the perspective of a performance architect, all bots exert a cost on the system, consuming finite resources that are best allocated to human users and completing their business processes.

Since 2014, the percentage of human users detected in the Imperva bot survey has risen from 40.9% to 57.7% in 2021. At the same time, the percentage of activity associated with bad bots has risen from 22.8% to 27.7% of observed traffic. Even 'friendly' bots can distort performance significantly. In 2014, a cross-shopping bot produced by Result.ly began proving prices on the main website for QVC. The load from the friendly bot took QVC.com offline for two days. QVC responded by taking the manufacturer of the bot solution to court.[2]

On a performance engineering basis, we recommend blocking bots. It is far less expensive to tailor bots with firewall rules than to purchase extra capacity or improve the efficiencies of code to overcome the resource pool locked by bots in the system. This begs the question, "How does one recognize a bot on the site? What are the signatures to be looking for?"

Identification of Bots falls into one of six different categories:

Walkers These are typically associated with product sites. They will begin with a product ID 1, index the product page, and walk page after page. Decommissioned or unavailable items will result in a high number of 404 errors for missing pages. The "tell" for this becomes a combination of odd navigation behavior plus a high number of 404s in a given session.

Spiders/Crawlers These are the classical site indexing robots, but nasty crawlers exist, which are both nefarious (examining returned code for known exploits) and simply poorly designed for operating at a too high a frequency, as with the Result.Ly/QVC example above. The tell here involves a combination of elements. Always leave some non-visible tags on a page that navigate to a page not reachable otherwise. Good bots will respect robots.txt designations. Bots that are trying to hide will

[1] https://www.imperva.com/resources/resource-library/reports/bad-bot-report/

[2] https://casetext.com/case/qvc-inc-v-resultly-llc

ignore robots.txt and will eventually hit the hyperlinked destination, which is not reachable by actual site users.

Blasters The observable symptoms for these bots are frequency of request. Good bots will follow a "do no harm" strategy, introducing delays between requests that mimic those of an end-user, ranging from a few seconds to up to a minute before following an additional URL reference in the returned source code. Blasters will follow up on requests with no delays, which distorts performance. With many blaster-styled bots on a site, the effect is similar to that of a DDOS attack. The majority of the time, these types of bots are associated with spot sale events on eCommerce sites & venue ticketing sites.

Location-based Location-based bots tend to be well hidden in the system, often mimicking real user patterns. The tell is in the amount of load produced by the location over time; it is typically orders of magnitude larger than an actual user due to the substantially larger number of requests. These items also tend to originate from data centers, not end-user computers.

User-Agent Bots often contain odd user agent strings, containing references to scripting languages and tools such as "PHP," "Python," "Curl," "JavaScript," and, at times, the use of open-source testing tools, "Jmeter," "Grinder," etc...

GZIP Bots rarely ask for compressed results in the request header. Unless a developer has explicitly removed the reference, client originated requests rarely do not include GZIP acceptable compression. Make this mandatory for partners. It will become easier to tailor out the bots.

How do an organization respond once a bot is identified? The natural response is to say, "We'll just block them!" That is not always the best response to a bot. "Why?" Once a bot is discovered and blocked completely, the bot developer will seek to change the bot to get around the new rule. This could be a change in behavior, location, frequency, or other attributes which are keying the blocking rule. This constant change of the bot developer paired with new bot rules leads to an arms race where both the bot developer and the site owner are working in tandem to defeat the other's efforts.

Blocking cloud-based data centers where Virtual Machines can be rented on the cheap for bot hosting is always a good strategy, but for other bot types taking a psychological approach to managing bots works better. A question, "What frustrates developers attempting to troubleshoot code?" The answer, "Code that works sometimes, but not all of the time." Consider the use of two items for these non-location-based bots:

1. Have a "slow node" in your deployed application architecture where the application firewall can route the session. This node is no different than what a user would hit, except it is reserved for when a bot is dynamically identified. Steer the bot to this one node to remove the load from the primary site leveraged by end-users. It can

be OK for this site to go up or down, experience poor performance, and be removed from a scaling group. This node (or nodes) exists to serve the bots.

2. Random HTTP 503 for some time. HTTP 503 is a standard message to return to users when a site is unavailable due to high loads. It is an expected message when a location cannot respond to end-users in a timely fashion. Leverage this by randomly impacting bot-identified sessions for 503 for random periods of time at the most distant firewall node from the application core, such as the Web Application Firewall (WAF) included as part of CDN providers.

Maintaining performance is as much about removing load from the system which should not be there as much as it is in making sure the expected load performs at it's highest level. By identifying the types and behaviors of automated agents on your system we are able to provide rule definitions to block the agents long before they impact site performance. We examine your site and your requests to recommend rules to be implemented within your Content Delivery Network (CDN) or Firewall to keep the agents at bay.

Effective bot management results in the following:

- Increased system resources available to your real users on high-volume events such as announcements, sales, promotions, etc.
- Less competition from automated agents on hot sales items, resulting in a higher likelihood of a real user sale.
- Lower hardware utilization in both owned and cloud-provisioned servers. This will result in lower cloud hosting costs and increased lifespan of servers.

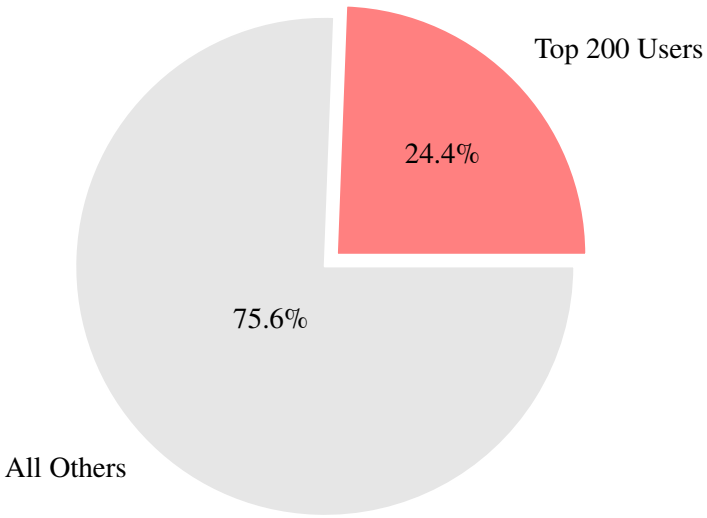


Figure 1.1: Load, Top 200 Users

To uncover the impact of bots, we carefully examine the load generated by the top 200 users of your public website. These users account for 24.4% of the load of your website. We examine both the characteristics of the load as well as the source of the load.

Rank	Load	IP Address Range	Source	Mitigation
1	5%	10.16.18.0/8	RoadRunner Hosting. A very aggressive spyder agent which is making calls to website with almost no delays between requests	This is a cloud hosting provider with no actual users present. Implement firewall rule to block address.
2	3%	10.168.72.0/16	Martian University of Belarus. Source of a variety of bots walking the sites at various times	As Acme Corporation does not do business in Belarus due to trade restrictions on product line, this address range can be firewalled off
3	2.5%	157.52.201.0/24	Global Frag Networks. This is the same hosting provider as Acme Corporation. Agents are crossing the backbone at the hosting data center for fastest possible access to the Acme Corporation website	From scamalytics.com, We consider Global Frag Networks to be a potentially high fraud risk ISP, by which we mean that webtraffic from this ISP potentially poses a high risk of being fraudulent. Other types of traffic may pose a different risk or no risk. They operate 9,106 IP addresses, almost all of which are running servers and anonymizing VPNs
4	...	...	...	...

Table 1.1: System Bots

Findings

The automated agents (Bots) used to generate load against ACME's eCommerce website fall into two areas of interest, location, and pattern.

15% of the total load comes from ten Internet subnets that are hosting a variety of Bot types that are impacting performance by overdriving the site. The IP Address Ranges are listed in the table 1.1 "System Bots" for mitigation in terms of overall load.

The remainder of identified Bot load is pattern based and requires a different set of rules for blocking.

During the last anvil spot sale, the number of users attempting to purchase anvils from the the southwestern United States jumped by 4000%, with a large number of the purchased anvils were later showing up on eBay and Craigslist. This is typical of Bot purchasing behavior and frustrates real users who are unable to exercise the system as fast as the automated agents, thus losing online sales

Upon examination, there were two classes of Bots that were impacting the purchasing at that time, an add-to-cart blaster Bot and a denial Bot that was constantly loading the page and the cart, impacting access to the end users.

The cart blaster Bot has the following characteristics:

- Thirteen-digit timestamp used as part of the object ID does not change across requests once the Bot starts
- The Bot hits the cart directly without loading additional page resources that an actual user loads

Mitigation: Implement a Firewall rule that looks for the following condition:

> RULE: *If three identical requests occur from the same IP address within a 15 second window against host mycart.acme.com, then block the user for a random period of time from ten to thirty minutes, responding with a 503, server busy.*

The denial of service Bot does not run JavaScript and, as a result, does not request the tracking object located on each page. An actual user loads the tracking object. As a result, we can use this lack of request to the tracking object as high confidence identifier of a request not from a user.

Mitigation: The tracking object exists on pages anvil1.htm, mycart.htm, hotitem.htm, and redrockets.htm. The rule should block the agent at the firewall:

> RULE: *If a user requests (anvil1.htm OR mycart.htm OR hotitem.htm OR redrockets.htm) and does not request (tracking.obj) block for a random period of time from ten minutes to thirty minutes responding with HTTP 503, service busy or unavailable.*

. . .

With these rules, we predict that 75% of the Bot load will be removed. This load represents overhead on the system that prevents the server from responding quickly to end-user requests, particularly during times of high load. This leads to faster client response times.

Chapter 2 Site Configuration, Issues impacting Response times and Performance

Independent of the Bot load, we'll take a look at the behavior of the users on the system for indications of how system configuration can be improved

Caching Strategy

The fastest response time is one not made across the network, where data is already satisfied at the localhost. The second most immediate is that resolved by an intermediate cache located near the client - This is commonly the role of the Content Delivery Network (CDN). And then finally, the slowest is a request which has to be satisfied uniquely by the back-end services of a system, for these requests involve arbitrated use of finite system resources for similar requests coming from other users of the system.

Outside of universal data, common data, which all users on a common scale require, there will be classes of data that fit into the following definitions:

- Data is static for the context of a user session. This can be session keys, security tokens, account information, etc...
- Data is common and static for a subset of users for some time. This data typically has age or time when it is no longer valid for use.
- Data, which is dynamic for every request. These are typically account-specific items, such as the request for a trade that would alter the stored state of the account.

For items tied to the session, we recommend a timeout model tied to the overall session timeout. Data should be marked as privately cacheable, with a fifteen-minute timeout from the first request. So, the headers would be as follows. Narrowing the window for session-related items to timeout on both client and server minimizes the window for a session hijack of a valid session as well.

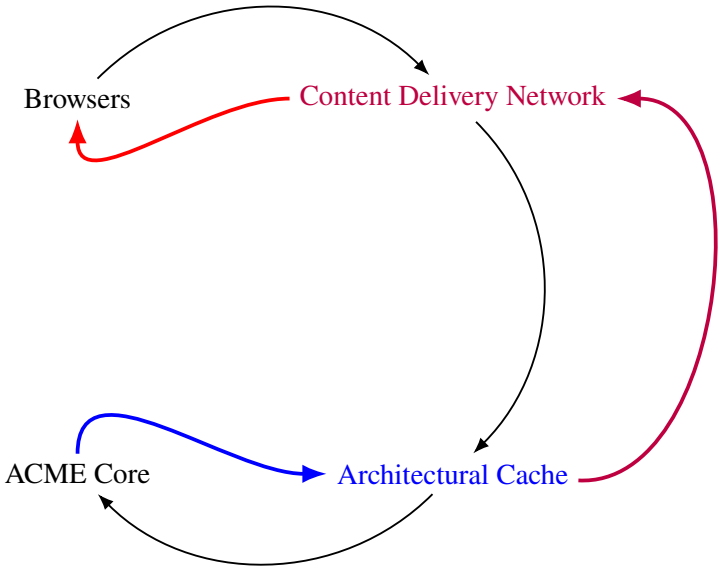
```
Cache-Control: private, max-age=900, must-revalidate
```

This includes the aging on the auth token, taking it from 24 hours to 15 minutes. This will significantly reduce the window for fraud with a valid, but not yet expired auth token. Any locked resources associated with not yet expired tokens are also freed for reallocation to live sessions.

For data that is shareable across users but where data has timeliness associated with the data, it is recommended that both an architectural cache in front of the system

(Varnish[1]/nginx/...) is leveraged as well as specific headers to allow for caching in downstream caches (Squid[2], Akamai[3], Fastly[4], etc...). Leveraging a CDN for a cache that is part of the application architecture is encouraged, as the multigenerational cache will reduce the load on the architectural cache.

In the diagram below, black lines represent requests, and colored lines responses from the system. The Architectural cache stores the ACME responses. The Content Delivery Network (CDN) then stores data served from the architectural cache. The net effect is that any request served from cached information at the client, from the cache in the CDN, or the architectural cache represents a request that the origin servers in the ACME core will not have to address.



These headers need to be well marked with the expiration of the data at a specific date and time. Although max-age is the preferred and recommended model over expires, given the latency of response for systems under load, a risk-averse path to ensure that stale data is not used is to leverage an explicit Expires header.

```
Expires: Tues, 09 Aug 2023 07:28:00 GMT
Cache-Control: public, must-revalidate
```

For the Dynamic requests, a simple header to ensure that no intermediate caches or client seeks to store the data.

```
Cache-Control: no-store
```

[1] https://varnish-cache.org/intro/index.html

[2] http://www.squid-cache.org/

[3] https://www.akamai.com/solutions/content-delivery-network

[4] https://www.fastly.com/products/cdn

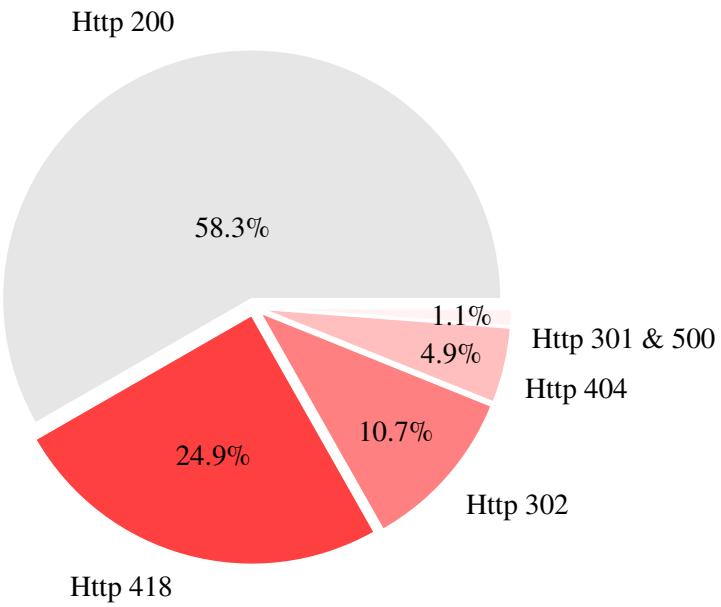


Figure 2.1: HTTP Status Responses

Every user calls the local Teapot service generating a 418 for every page visited. This represents a high overhead for the server to report the brewing status on each page. Fortunately, this is a cacheable event and can either be cached or served by the Content Delivery Network (CDN) or the client. As user sessions are short, at 15 minutes, this is best cached at the client for the duration of the user session.

The Acme Corporation uses CDN priority for CDN configuration, so all configuration instructions are biased to the CDN.

Status 418, Teapot Service, Mitigation

In the CDN Management console, set the CDN cache age for 418 responses to 820 Seconds with a matching client cache age. This should reduce the number of requests to the teapot service to one per-user session with a single revalidate. This improves client responsiveness in three ways:

1. For the majority of the user session, the request is serviced from the client cache without requiring a request and response from the server or CDN cached value.
2. The CDN will handle a single request and periodic validation for all of the servers connected to the CDN. This will remove a quarter of the load from the web servers required to handle the teapot service requests.
3. More resources on the server will be available to handle other requests under high load conditions, resulting in a more responsive server.

HTTP status 302, temporary redirect, must be handled by the web server for dynamic content. Errors, represented here by 404, and permanent redirects, represented by 301, can be cached by a CDN.

This CDN delivery of content improves performance by serving the content from the closest local network location for the duration of the session, avoiding network traffic, and removing redundant requests from the server. By shortening the network, response time to the client is improved.

Status 404, File Not Found, Mitigation

With a weekly release schedule, we will set the s-max-mage to 604800 seconds for 404 responses. This represents the amount of time between releases.

Set the client cache for 12 hours in the CDN management console. With a setting for the CDN for revalidation with the server at one week; the value will remain in cache for a week to serve users. Purging your CDN cache at the time of your release will ensure that any cached 404's that are no longer valid will not be passed to the end user.

This improves end-client performance by:

1. Servicing the 404 from the client for the duration of the typical user session and then from the CDN node closest to the user in between releases. This minimizes the network overhead of requesting a file from the source web servers.
2. The load of the 404 requests is removed from the server almost completely. Only a minimal number of 404 requests will remain, which at maximum will be the number of missing resources multiplied by the number of CDN servers engaged by users. More resources are available to dedicate to direct requests. Error logs will also shrink.

TIP:Developers should reply to application errors with an HTTP 200 with an embedded error message. HTTP 404 is reserved for a missing resource associated with a web request

HTTP 301, Permanent Redirect, Mitigation

Redirects are expensive. The server has to respond to the original request with a location to a new file, and then the client has to make the re-request. By offloading this request to the CDN, the target for the permanent redirect can be requested, avoiding a second request for information

Within the CDN console . . .

Session Persistence

Timeout represents the length of time to hold session data on the server and lock resources associated with a session after the last request from the client. So long as the end-user continues to engage with the application and the back-end host for dynamic data, the session should remain open indefinitely. An aggressive timeout is recommended for high volume, high use systems for those who go idle.

The classic example of what happens when resources are held for long periods is a very public one, Healthcare.gov. High user loads and long session timeout resulted in systems running out of resources to serve users. Users were compelled to spend hours filling out forms and then submitting them for a quotation. Only six people received quotes on services on the first day of operation, with the remainder locked out due to system issues.

The lock window for a high user, high transaction system should be as small as possible. Resources are finite. Any resource locked and waiting to timeout is a resource that should be re-assignable to a new session. The current 24-hour timeout ensures that ACME's servers continue to lock resources for at least an hour beyond ACME's peak system load within a day.

This data set is derived from information captured in timestamps, IP/Session, requests, and REFERER fields of the log data. From this data we can objectively pull the time between top level page requests for an individual session. This is often called dwell time, or the time between requests when a user is on a page. Where session timeout is greater than the maximum observable dwell time system resources related to session are being held too long after a user departs a site.

The current session policy for web servers holds user sessions for 24 hours. The maximum page-page-transition time over the 30-day sampled window of HTTP access logs is 10 minutes, with the 99th percentile at 8:32 and the 50th percentile at 3:00.

In short, the session duration is set too long for the behavior of the users. The session is holding the user object based resources too long before release. This leads to resource starvation on the web/application servers for high load events for Acme Corporation.

TIP:It is less costly to the server resource pool to re-allocate a resource for a returning user after the fifteen minute period than it is to hold resources for all users for an additional twenty-four hours

Adjust the session timeout from 24 hours to a maximum of 15 minutes, five minutes beyond the longest observable page-to-page transition. This will free all user object based resources nearly a full day earlier than previously, with those resources being available for new users to the site. For your web server information on how to modify the default web session timeout is located . . .

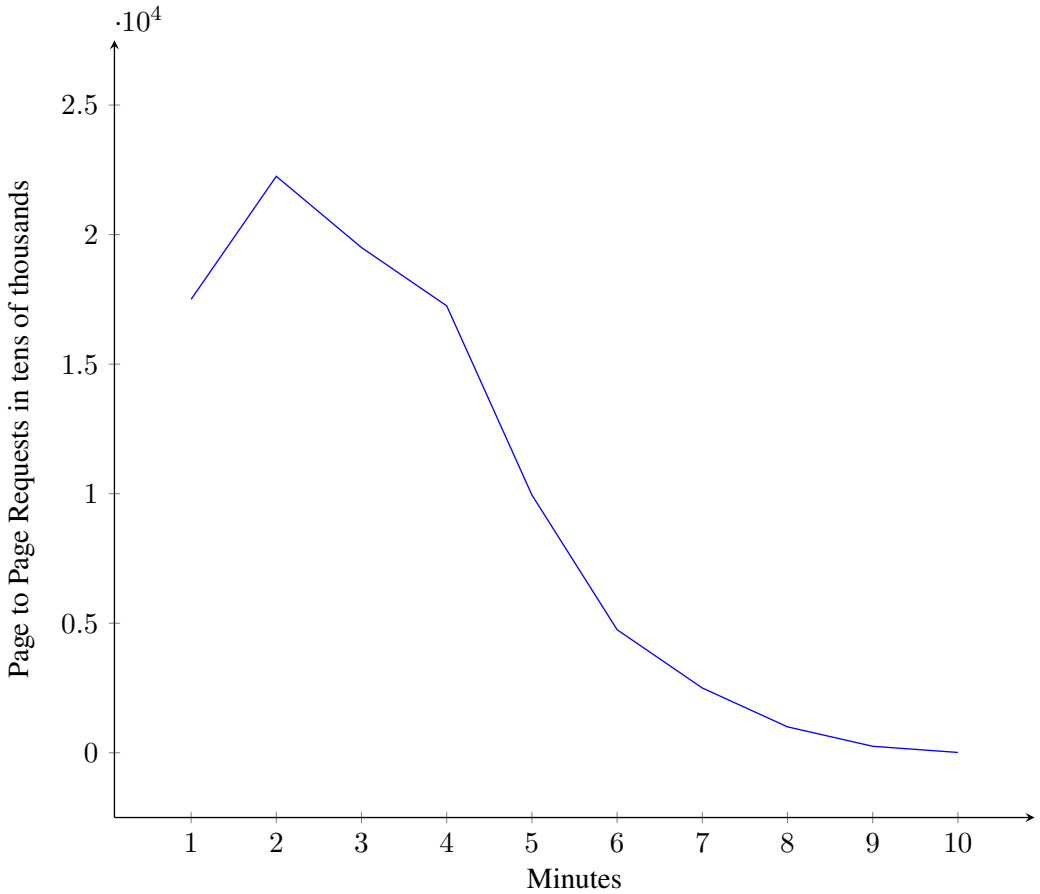


Figure 2.2: Page to Page Transitions. Source:HTTP Logs

Chapter 3 Cart Subsystem

Shopping cart examination is broken down into four categories: Cart Allocation, Cart Deallocation (recovery), Cart Persistence, and Aging Policies. An inefficient cart system can degrade performance in several ways:

- Too many resources can be allocated to non-converting users. This causes resource starvation for users on a conversion path to revenue. The result is a slower response time with high abandonment. Spot sales can lead to resource exhaustion for the entire cart subsystem.
- A long path to resources. Too early allocation of a shopping cart causes the end user to maintain a relationship with dynamic elements of the cart subsystem behind the content delivery network (CDN) tier. This results in a slowed page asset delivery, a longer render time and a higher abandonment rate.
- Storage costs. Improper storage of abandoned carts leads to an ever-escalating need for persistent cart storage. Not only does this increase the cost of physical storage to cover the carts, but it also increases the traversal time for the cart storage system to retrieve a valid cart. Both financial and performance penalties result.
- Product orphans. Older carts may contain items that are no longer offered by the company for purchase, either because a product is no longer in production or due to changes in partner relationships. Orphaned products must be removed from carts to free up storage and improve performance.

Cart Allocation

One of the common problems in scalability is too early allocation of the shopping cart. An examination of the cart allocation process shows that the cart is not allocated before the end client needs it. There are no changes required in the cart allocation model.

Cart Deallocation

Cart deallocation policies are aligned with *Session Persistence* for the website under test. A maximum cart timeout of fifteen minutes is recommended, a reduction of twenty minutes from the current cart timeout of thirty minutes. Please review the previous chapter on session persistence.

Cart Persistence

A persistent cart allows users to store information in the cart that can then be accessed in future user sessions. The goal is to convert users to purchasing users by reminding them of the items in their carts.

An examination of the cart storage system resulted in the following profile of stored carts:

Cart Items stored	#Carts
0	56,725,612
1	622,016
2	79,486
3	18,932
4	7,716
5-10	3,601
11-50	1,829
51-100	502
101-500	276
501++	12

The maximum cart had over 12,000 items in the cart. Every time this cart is loaded and unloaded, the cart management system is impacted for all system users, slowing cart access. This will be most noticeable during periods of high loads, such as spot sales on Anvils and Rocket motors.

There are several items to address regarding carts:

TIP:Removing empty carts shortens the index traversal time required to pull a cart from storage which does contain items for a returning visitor

1. Acme Corporation is currently storing empty carts. This is leading to an acceleration in storage costs and degrading the amount of time required to both store and retrieve carts for existing users. This degrades response times. It is recommended that a SQL Script be executed just before the database backup window to remove carts with zero elements from storage. Over the long term, developers should alter the code to check for a zero-content cart before storage.
2. Cart policies should result in a daily email to customers where a cart exceeds 50 items in the cart.
3. Where carts have more than 100 items, customers should be called to facilitate the conversion or deletion of the cart.
4. Recommend an account manager for the owner of the largest cart,***Wile E. Coyote*** of Elephant Butte/Monument Valley, Arizona. The overhead of this one cart impacts all other users on the site when it is loaded for use.

Cart Aging

Outside of the zero content carts, 42% of the remaining carts have not been accessed in over six months. It is recommended that the following cart policies be implemented to address non-converting stored carts: . . .

Chapter 4 Custom Application Examination

There are common characteristics shared by high-user, high-transaction rate systems that are responsive to end-users under load. On the finite resource front, these systems seek to minimize the lock on resources used to satisfy end-user requests, aggressively trimming items related to sessions. They reserve the use of resources for business flows associated with revenue or mission. User transactions fall into three categories:

Informational	Conversion Path	Mission Completion
This information is familiar to all users as it is served primarily out of the cache. Examples include broadcast data such as stock trade tickers, landing pages & product information pages served entirely out of cache. The governing principle is that no finite server resources shared with conversion business processes are used for information delivery.	This category of business functions is part of conversion but is not the end process itself. Moving off of the cache, users can gradually access dynamic data as part of a mission or conversion. This might involve an inventory lookup to the current level as a part of a sale, a current balance lookup to close an account, or to obtain an instant quote for a stock trade.	The most significant resource allocation is associated with completing a business process. These items are often tied directly to revenue or, in a non-revenue situation, to completing a business process to support the enterprise mission. The execution of these functions is often connected to specific servers to avoid resource arbitration with non-task-related functions. Examples: Single click checkout on Amazon, utility payment completion, executing floor trade, or reporting a utility outage.

Not all users are equal in revenue, value, or system resources. Nor should they be treated equally within the system. Most low-performing systems' death knell is the allocation of resources to users who fall into the informational category. Typical examples include a default shopping cart for every user, when less than 20% of users on an eCommerce site begin the conversion process by adding an element to the shopping cart; healthcare.gov forcing users to fill out significant information before an informational quote (failed to recognize that users touch a site six times on average before beginning the conversion process); an always dynamic lookup for power outages when cached data of one to five minutes is sufficient.

Acme Corporation leverages a custom application for their eCommerce site. Both partners and end customers leverage this custom application for the purchase and management of ACME products, customer account information, management of invoices,

as well as open purchase orders. ACME's entire product catalog is also online, including the full line of Anvils, Explosives, Roller Skates, & Rocket Motors. Strategic customers, such as Warner Brothers Animation, leverage the just-in-time delivery partner portal.

A Profiler View Of The System

One of the most common questions in information technology is, "Why are we slow?" Invariably this comes down to where the weight, or the most significant use of resources, is concentrated within any given system. A profiler is the most common way to find where this weight exists. The most common architectural system profiler used for years is a database profiler. Database profilers track the total time or cost of a set of queries for the express identification of the most collectively expensive query. These queries are typically resource-bound, accessing a substantial amount of unindexed data, involve complex/expensive subqueries, or just flat return too much from the disk to the network and onto the client. Reducing the resource cost of the most costly query improves the one item being tuned and the system's overall efficiency by reducing the system waits associated with second, third, and subsequent queries and their executions.

Profiling has moved upstream to the application server's domain in the past decade. Tools such as Dynatrace, App Dynamics, NewRelic, and others have provided interfaces to the .Net and Java virtual machines to easily profile the highest-cost items running in the virtual machine. There is a cost associated with implementing these solutions, both in resource cost on the servers being profiled, a steep (but short) learning curve, and licensing.

QA Consultants has extended this profiling model to the web tier. This low-cost, passive solution requires only a time measurement to be included with each HTTP request. This allows for a roll-up of statistical information from the system for the highest cost requests and, thus, where to begin optimization for the most significant benefits.

Vilfredo Pareto, civil engineer. 1848-1923

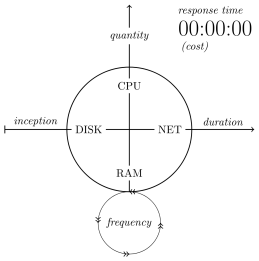
80/20 Rule

The Pareto Principle, commonly known as the 80/20 rule, holds that 80% of the consequences come from 20% of the causes. This has applications in two distinct areas of software performance engineering. The first is single-user performance. The effort needed to architect, design, and build a system that meets performance requirements for a single user will satisfy 80% of performance cases, with 20% remaining on the server side involving competition for resources in multiple user access. Leveraging early data to identify and solve for the 80% of performance cases allows for the exploration of more complex performance issues involving multiple user access.

The second application involves server code which is resource inefficient. The top 20% of inefficient code draws down system performance for the remainder of the system. The easiest way to illustrate this condition is by discussing database queries. A particular

query executed by multiple users without the support of an index can degrade the entire performance of a database server, even for those queries supported by an index. Why? Without an index in place, the query must be backed with a table scan, an exercise that examines every single row of a table without regard to table size to satisfy the query. The competition for access to the drive for reading and writing is degraded during this period for ALL users while the table scan is satisfied. The more ongoing scans caused by multiple users, the higher the impact. Optimizing the query with an index reduces the demand on the disk for an individual query, resulting in less of a resource block for all remaining queries. Database efficiency overall has improved.

Where is the bottleneck? Which component is the most expensive? Which selections of code do we optimize to minimize resource constraints and enhance scalability? As part of this site examination, the data gathered from the logs have been used to rank the top-level page requests by system cost. In addition to total system cost, the distribution of response time data has been examined to understand the resource sensitivity for individual requests. Both total cost and resource sensitivity have been used to rank the requests on the website. It is recommended that the top 20% of requests be profiled for additional code improvements in order of rank.



Request	Score	Rank	% Load
/account/services/rewardsprogram/updaterewards	6,725.612	1	23.2
/partner/services/licenses/api/v1/license_status	933.016	2	3.8
/catalog/servlets/view/UI/Prod	779.582	3	3.2
/account/services/account?lang=2	518.912	4	2.2
/account/services/credit/creditapplication	477.716	5	1.99
/catalog/servlets/Applet/SearchAnvils	293.691	6	1.2
/partner/services/transportcertificates/api/v1/csr_count	198.649	7	0.8
...	...	...	

Updaterewards is called on every single page once a person signs in, which places it's collective cost on the system quite high, hence the rank for improvement.

The Engagement

It all begins with data, the capture of data for analysis. Where Performance Testers concentrate their efforts on using a particular tool, Performance Engineers and Architects are concerned with data. Within that data, asking what patterns tell a story of whether exceptional or poor performance is present. How did we get to the data needed for this report? Logs!

Logs are collected in and analyzed in Splunk[1]. There are a minimum number of fields we need to have in place for analysis:

Date/Time Often, this is a 13-digit timestamp representing the number of milliseconds since January 1, 1970

Request Less any parameters (after the ?) which may include PII/PHI data

Method GET, POST, HEAD, . . .

HTTP Status 200, 301, 304, 404, 503, . . .

User Agent This field includes browser type (IE, Chrome, ...), platform type (Mobile, Tablet, ...)

REFERER Which top-level request (i.e., which page) is this request a component of?

cs-bytes Number of Bytes sent from client to server

sc-bytes Number of Bytes sent from server to client

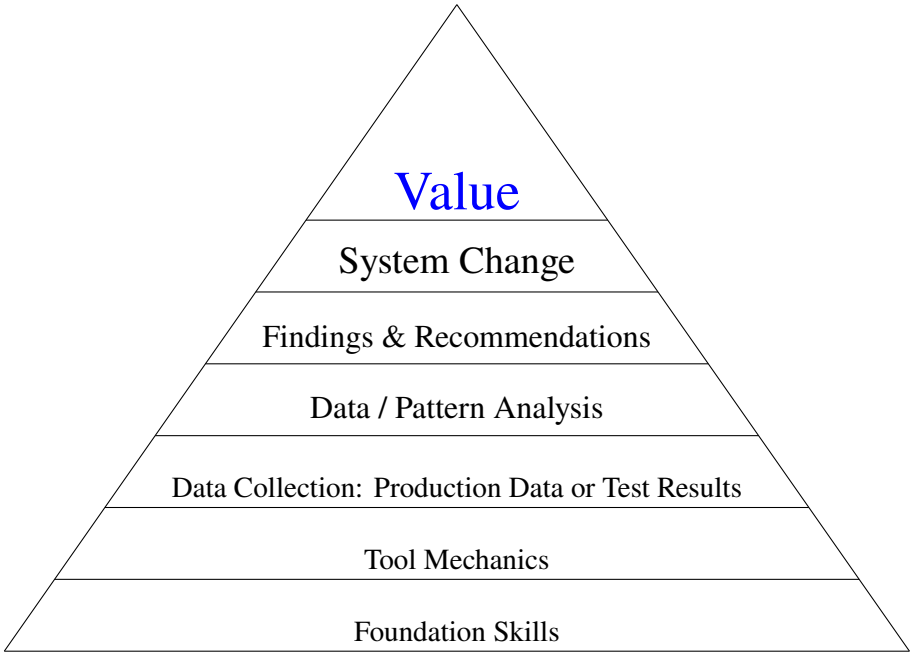
time-taken Time (in milliseconds) to satisfy the request

We recommend a minimum of one week of logs for analysis. Ideally, a 30-day window of logs is available for analysis.

Where information about software and versions is available for your application, please provide this data. This will provide a path to the admin console locations to make changes or mitigate issues. For cloud-hosted applications, access to information on your current cloud bill is required.

With the data in place, analysis becomes the responsibility of QA Consultants' Architects. The Architects benefit from a minimum of two decades of performance engineering and testing expertise, allowing them to analyze the log data for patterns related to poor performance. Patterns are complex, ranging from architecture to server configurations, even including programming models. The patterns identified lead to specific recommendations. Value is realized after changes have been implemented as a result of findings.

[1] https://www.splunk.com/



Value Dependencies in Performance Engineering

This documentation was prepared using Overleaf[2], a hosted version of the typesetting language LaTeX. Overleaf is a solution for multi-author use and producing technical documents.

[2] https://www.overleaf.com/