Experience Commerce Performance White Paper

Sitecore Experience Commerce 9.2.0

Sitecore Experience Commerce 9.2.0 Performance Testing

Table of Contents

Executive Summary 4
Content Delivery 4
Key Findings 5
Content Management 5
Content Delivery 5
Commerce Engine 6
Storefront 6
Deployment 7
Marketplace Settings 7
Configuration Scaling 8
Azure Cost 9
Monitoring Configuration 11
Additional Configurations 11
Methodology 15
Site Specification 15
Test Scenarios 15
Load Matrix 17
Data Configuration 18
Test Infrastructure Configuration 18
Performance Metrics 18
Results 20
Key Metrics 20
CPU – App Service Plan 22
App Memory 23
Threads 24
Azure Search 25
Azure SQL Databases 26
Azure Cache – Redis 26
Azure Cache – Commerce Redis 27
Appendix 28
Thread Starvation Hotfix 28
Private Session Processing 30
Shops ARR Affinity 31
Minions Wake-Up Interval 33

© Copyright 2019, Sitecore - all rights reserved.

Experience Accelerator Theming 34

Sitecore® is a registered trademark. All other brand and product names are the property of their respective holders. The contents of this document are the property of Sitecore. Copyright © 2001-2019 Sitecore. All rights reserved.

© Copyright 2019, Sitecore - all rights reserved.

Executive Summary

This white paper describes Sitecore Experience Commerce (XC) 9.2.0 performance testing. The main goal of the performance testing was to establish a performance baseline based on an out-of-the-box Large Tier XC deployment from the Azure Sitecore Marketplace. The tests focused on the following three main domains:

- Content Delivery
- Shops
- Analytics

Metric	Deployment	Value
Avg Response Time (sec)	WP-Base	0.65
	WP-HF-Best	0.55
Reqs/Sec	WP-Base	37.2
	WP-HF-Best	39.8
Avg Page Time (sec)	WP-Base	1.75
	WP-HF-Best	1.43
Pages/Sec	WP-Base	18.0
	WP-HF-Best	19.3

Content Delivery

Testing ran with a 5-minute ramp-up from 1 to 70 users and four hours of traffic at 70 constant users. User Think Time utilized a normal distribution profile based on two seconds.

The testing did not request static files. In a typical enterprise deployment, static content is in a Content Delivery Network (CDN) or cached by a network appliance.

The primary testing compares and contrasts two deployment scenarios:

- **WP-Base**: Default XC 9.2 Large Tier Sitecore Marketplace Deployment + Shops ARR Affinity=Off
- **WP-HF-Best**: Default XC 9.2 Large Tier Sitecore Marketplace Deployment + Shops ARR Affinity=Off + hotfix + performance tunings

> **Note**
> The hotfix in the second deployment scenario was created as a result of the testing for this white paper.

© Copyright 2019, Sitecore - all rights reserved.

Key Findings

Content Management

With 500K users with addresses configured, the aspnet_Users, aspnet_UsersInRoles and aspnet_Membership tables in the Core DB were highly fragmented. Querying for the row count took in excess of 30s and the User Manager portal was inaccessible due to a SQL timeout. To improve query performance, we reorganized these indexes. After reorganization, the queries were returning within 1s.

```
ALTER INDEX ALL ON [dbo].[aspnet_Users] REORGANIZE
ALTER INDEX ALL ON [dbo].[aspnet_UsersInRoles] REORGANIZE
ALTER INDEX ALL ON [dbo].[aspnet_Membership] REORGANIZE
```

Notes

- *Reorganization of each index took approximately 25m.*
- *After reorganization, the User Manager portal was still inaccessible.*
- *Only after the Core DB was scaled to P1 did the User Manager portal become accessible again.*

Content Delivery

Thread starvation issues due to session end processing were encountered while testing the WP-Base deployment. As a result, a hotfix was produced for this issue and was consequently verified in the WP-HF-Best deployment. (See the appendix – *Thread Starvation Hotfix*). The thread starvation issues caused all the CD instances to restart.

The site can get into a situation where the Platform Redis memory/keys suddenly start increasing steadily over time and end up surpassing allotted memory limits. The exact cause is unknown, but what was observed is that the issue is correlated to the session end processing of private sessions. The steady rise in memory/keys appears to start when there is a divergence in the number of shared and private session keys in the Redis databases.

Note
To be able to monitor these values, the shared and private sessions were written into separate databases db0/db1).

As time progresses, the session end processing for shared sessions appears to be able to keep up. However, the private session processing lag gets progressively worse until the allotted Redis memory is exhausted. Once it reaches this point, the throughput on the all the CDs are reduced drastically to a few reqs/sec. (See the appendix - *Private Session Processing*).

Based on the deployments in these tests, tuning the pollingInterval=30 and pollingMaxExpiredSessionsPerSecond=20 appeared to reduce the likelihood of the issue from occurring.

The site can get into a situation where private sessions remain in the queue forever. A fix from the development team was successfully verified, but a hotfix was not available at the time when this report was created.

© Copyright 2019, Sitecore - all rights reserved.

Note
The fix was in Sitecore.SessionProvider.Redis.dll.

Commerce Engine

Compared to XC 9.0.3 performance testing, disabling the ARR Affinity of the engine resulted in much more even distribution of load among the instances, and consequently better average throughput and response times. This was due to request spikes that appear to be isolated to a single shop instance, triggered by cache repopulation. (See the appendix – *Shops ARR Affinity*).

Notes

- *Setting the Shops ARR Affinity=Off improves the request distribution, but some request spikes remain*.

  The size of the Shared DB increased rapidly.

- After the 100K Catalog and 500K users were deployed, the Shared DB size was already at 75GB.
- After a little over a month of sporadic testing (test durations ranging from 4hrs to 18 hrs), with an average user traffic rate of 70 users/sec (at 2s Think Time) performing browsing and checkouts, the size of the Shared DB grew to over 250GB, surpassing the Large Tier default deployment limit. The four largest tables, according to size, were: CommerceEntity (>182GB), OrdersEntity (>38GB), CustomersEntity (>14GB), and CommerceLists (>4GB).
- The four largest tables mentioned above were over 99% fragmented and there were numerous others near the same level of fragmentation or over 90%.
- Reorganization of the indices was performed instead of a rebuild to reduce the impact (Data I/O) and execution time. Even with the reorganization, it took 4h10m to complete the CommerceEntity table indices, while the other three took up to 55m each. As such, the reorganization could not be performed from the Azure Query Editor interface, since the client timeouts for those are set too low. An external connection from SQL Management Studio to Azure SQL was required in order to complete the index maintenance.

The wake-up interval is set to 5m for all the minions, except for “Carts”. This causes Minion App Server/SQL CPU, SQL Connection, and SQL Data I/O spikes during those intervals. For improved stability and performance, processing should be distributed as evenly as possible, minimizing processing spikes. As such, it is recommended that the wake-up intervals be tuned to allow for a more even distribution. (See the appendix – Minions Wake-Up Interval).

Storefront

The Experience Accelerator Theming Optimiser Scripts/Styles mode is disabled by default. These should be set to “Concatenate and Minify” to reduce data throughput and improve response times. (See the appendix – *Experience Accelerator Theming*).

© Copyright 2019, Sitecore - all rights reserved.

SITECORE

Deployment

Marketplace Settings

Version: 9.2 Initial Release

Topology: Sitecore Experience Cloud (XP)

Configuration: Scaled Production

Size: Large

Additional Modules:

- Sitecore Commerce 9.2 Initial Release
- Sitecore Experience Accelerator Storefront 3.0 Initial Release
- Sitecore Experience Accelerator 1.9 Initial Release

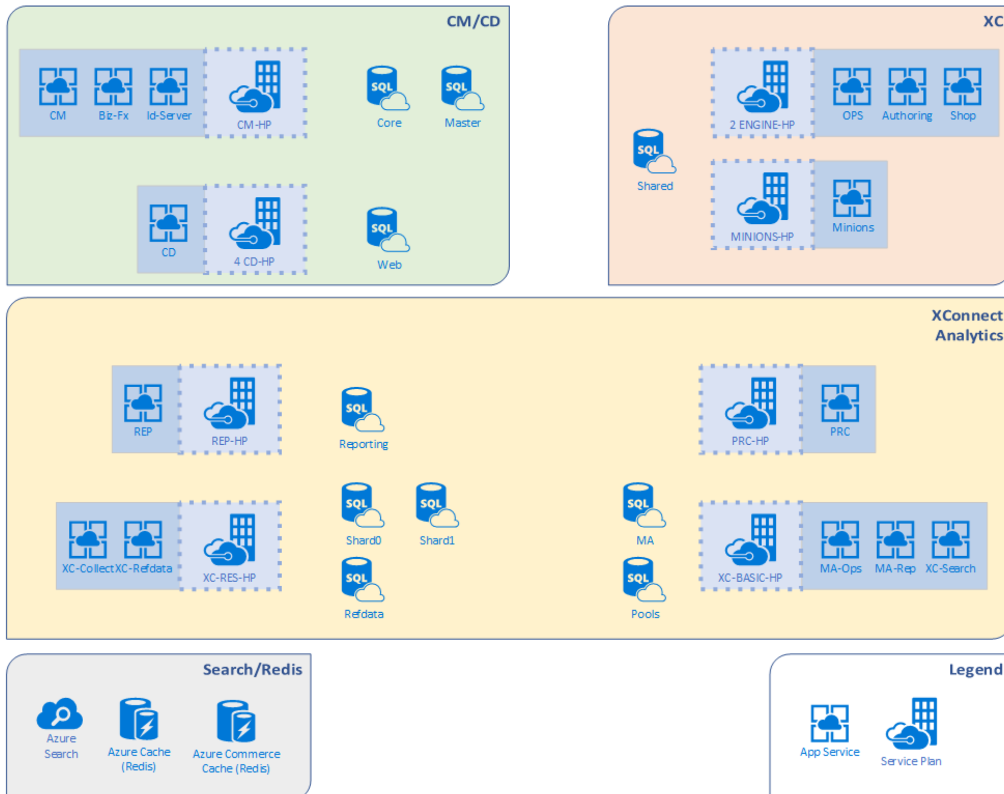
Search Provider: Azure Search

Search Replica Count: 1

Location: East US

Deploy a dedicated dispatch server for EXM: False

Use Application Insights: True



© Copyright 2019, Sitecore - all rights reserved.

Configuration Scaling

App Service Plan

App Service Plan Name	Scale	Instance Count
cd-hp	P3v2	4
cm-hp	P3v2	1
engine-hp	P3v2	2
xc-resourceintensive-hp	P3v2	1
xc-basic-hp	S3	1
minions-hp	S1	1
prc-hp	S2	1
rep-hp	S2	1

Databases

Database Name	Scale
shard0-db	P2
shard1-db	P2
web-db	P1
shared-db	Gen4, 3vCores, 350GB
core-db	S1
master-db	S1
pools-db	S1
refdata-db	S3
ma-db	S1
reporting-db	S2

Azure Search

Name	Units
as	S1x3 = 1 Replicas, 3 Partitions

© Copyright 2019, Sitecore - all rights reserved.

Azure Cache – Session State Redis

Name	Scale
redis	C1

Azure Cache – Commerce Redis

Name	Scale
redis	C3

Azure Cost

Disclaimer

The cost estimates in the following table are based on this document at the time of writing. Furthermore, the estimates only apply to the services detailed in the Configuration Scaling section.

Microsoft Azure Estimate			
Your Estimate			**@ 730 Hrs**
Service type	**Region**	**Description**	**Estimated Cost**
Azure Cache for Redis	East US	C1: Standard tier, 1 instance(s), 730 Hours	$100.74
Azure Cache for Redis	East US	C3: Standard tier, 1 instance(s), 730 Hours	$328.50
Azure Cognitive Search	East US	Standard S1, 3 Unit(s), 730 Hours	$735.84
App Service	East US	Premium V2 Tier; 8 P3V2 (4 Core(s), 14 GB RAM, 250 GB Storage) x 730 Hours; Windows OS	$4,672.00
App Service	East US	Standard Tier; 1 S1 (1 Core(s), 1.75 GB RAM, 50 GB Storage) x 730 Hours; Windows OS	$73.00
App Service	East US	Standard Tier; 2 S2 (2 Core(s), 3.5 GB RAM, 50 GB Storage) x 730 Hours; Windows OS	$292.00
App Service	East US	Standard Tier; 1 S3 (4 Core(s), 7 GB RAM, 50 GB Storage) x 730 Hours; Windows OS	$292.00
Azure SQL Database	East US	Single Database, DTU Purchase Model, Premium Tier, P2: 250 DTUs, 500 GB included storage per DB, 2 Database(s) x 730 Hours, 5 GB Retention	$1,825.00
Azure SQL Database	East US	Single Database, DTU Purchase Model, Premium Tier, P1: 125 DTUs, 500 GB included storage per DB, 1 Database(s) x 730 Hours, 5 GB Retention	$456.25
Azure SQL Database	East US	Single Database, DTU Purchase Model, Standard Tier, S1: 20 DTUs, 250 GB included storage per DB, 4 Database(s) x 730 Hours, 5 GB Retention	$117.74
Azure SQL Database	East US	Single Database, DTU Purchase Model, Standard Tier, S2: 50 DTUs, 250 GB included storage per DB, 1 Database(s) x 730 Hours, 5 GB Retention	$73.61
Azure SQL Database	East US	Single Database, DTU Purchase Model, Standard Tier, S3: 100 DTUs, 250 GB included storage per DB, 1 Database(s) x 730 Hours, 5 GB Retention	$147.18

© Copyright 2019, Sitecore - all rights reserved.

Azure SQL Database	East US	Single Database, vCore Purchase Model, General Purpose Tier, Provisioned, Gen 4, 1 3 vCore instance(s) x 730 Hours, 32 GB Storage, 0 GB Backup Storage	$555.96
		Monthly Total	**$9,669.81**
		Annual Total	**$116,037.76**

© Copyright 2019, Sitecore - all rights reserved.

Monitoring Configuration

The default monitoring configuration from the out-of-the-box Sitecore Marketplace Large Tier XC.

Additional Configurations

Deployment Scenario 1: WP-Base

The deployment is based on a default XC 9.2 Large Tier Sitecore Marketplace deployment, as specified in the "Deployment" section, except for the following:

ARR Affinity

The ARR affinity for the Shop was set to *Off* to improve performance of the stateless app by allowing a more even distribution of load among the instances.

App Service: Shop

Shop, Configuration, General settings

ARR affinity = Off

Deployment Scenario 2: WP-HF-Best

The deployment is based on a default XC 9.2 Large Tier Sitecore Marketplace deployment, as specified in the "Deployment" section, except for the following:

ARR Affinity

The ARR affinity for the Shop was set to *Off* to improve performance of the stateless app by allowing a more even distribution of load among the instances.

App Service: Shop

Shop, Configuration, General settings

ARR affinity = Off

Session State Providers

To optimize the processing of the session state providers and achieve a balanced steady state for long traffic durations, compression was enabled and the maxConcurrency, pollingInterval, and pollingMaxExpiredSessionsPerSecond were tuned.

App Service: CD

File: web.config

```
<add
```


© Copyright 2019, Sitecore - all rights reserved.

```
    name="redis"
    type="Sitecore.SessionProvider.Redis.RedisSessionStateProvider,
Sitecore.SessionProvider.Redis"
    applicationName="private"
    connectionString="redis.sessions"
    polingInterval="30"
    databaseId="1"
    compression="true"
    maxConcurrency="800"
    pollingMaxExpiredSessionsPerSecond="20" />
```

File: App_Config/Sitecore/Azure/Sitecore.Analytics.Tracking.Azure.config

```
<add
    name="redis"
    type="Sitecore.SessionProvider.Redis.RedisSessionStateProvider"
    connectionString="redis.sessions"
    applicationName="shared"
    operationTimeoutInMilliseconds="5000"
    pollingInterval="30"
    compression="true"
    maxConcurrency="800"
    pollingMaxExpiredSessionsPerSecond="10"
    patch:after="*[1]" />
```

Commerce Engine Connect Memory Cache Settings

To optimize the Commerce Engine Connect memory cache, the cacheMemoryLimitMegabytes and cacheExpiryInSeconds were tuned to use more memory and have a longer expiration period.

App Service: CD

File: App_Config/Include/Y.Commerce.Engine/Sitecore.Commerce.Engine.Connect.config

```
<cacheMemoryLimitMegabytes>1500</cacheMemoryLimitMegabytes>
<cacheMemoryScaleFactor>3</cacheMemoryScaleFactor>
<physicalMemoryLimitPercentage>0</physicalMemoryLimitPercentage>
<pollingInterval>00:03:00</pollingInterval>
<cacheExpiryInSeconds>14400</cacheExpiryInSeconds>
```

Content Search Logs

Reduce the logs generated by setting the log levels to WARN.

App Service: CD

File: App_Config/Sitecore/ContentSearch/Sitecore.ContentSearch.config

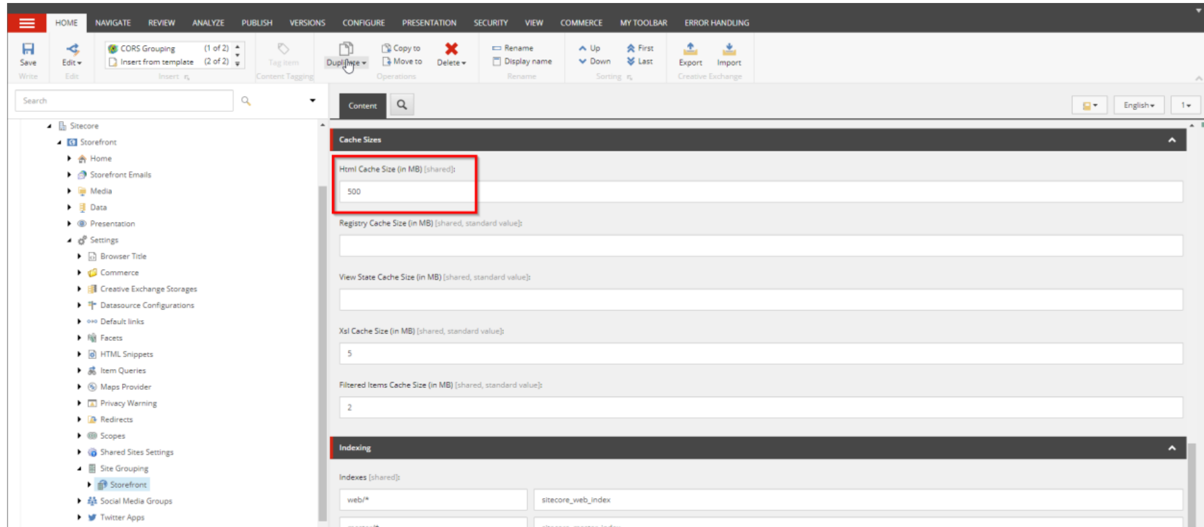
```
<logger name="Sitecore.Diagnostics.Search" additivity="false">
    <level value="WARN"/>
    <appender-ref ref="SearchLogFileAppender"/>
</logger>
<logger name="Sitecore.Diagnostics.Crawling" additivity="false">
    <level value="WARN"/>
    <appender-ref ref="CrawlingLogFileAppender"/>
</logger>
```

Storefront Cache Size

Increase the Html Cache Size of the Storefront using the Content Editor.

© Copyright 2019, Sitecore - all rights reserved.

Content Editor



Sitecore Cache

Increase the capacity of various caches.

App Service: CD

File: App_Config/Sitecore.config

```
<!--  ACCESS RESULT CACHE SIZE
            Determines the size of the access result cache.
            Specify the value in bytes or append the value with KB, MB or GB
            A value of 0 (zero) disables the cache.
-->
<setting name="Caching.AccessResultCacheSize" value="500MB" />
```

```
<database
    id="web"
    singleInstance="true"
    type="Sitecore.Data.DefaultDatabase, Sitecore.Kernel"
    role:require="Standalone or ContentManagement or ContentDelivery">

    <cacheSizes hint="setting">
        <data>500MB</data>
        <items>500MB</items>
        <paths>2500KB</paths>
        <itempaths>100MB</itempaths>
        <standardValues>2500KB</standardValues>
    </cacheSizes>
</database>
```

```
<!--  CACHING - CACHE KEY INDEXING ENABLED - ACCESS RESULT CACHE
Determines whether or not the system uses extended indexed storage for
the cache keys of the AccessResultCache.

Using indexed storage for cache keys can in certain scenarios
significantly reduce the time it takes to perform partial cacheclearing of the
AccessResultCache. This setting is useful on large
solutions where the size of this cache is very large and
where partial cache clearing causes a measurable overhead.

However, enabling this setting on content management servers with many
```


© Copyright 2019, Sitecore - all rights reserved.

```
editors and many content items can degrade performance.

Default value: false
-->
<setting
        name="Caching.CacheKeyIndexingEnabled.AccessResultCache"
        value="true" />
```

Session Provider Hotfix

The following CD App Service hotfix was applied to address thread starvation issues:

- https://kb.sitecore.net/articles/673673

© Copyright 2019, Sitecore - all rights reserved.

SITECORE

Methodology

Site Specification

Configuration	Values
Web technology	ASP.NET MVC, .NET Core
Number of pages	Over 100K (product detail pages)
Multi-language	4 languages
HTML caches are used	True
Number of user profiles	500K
Starting Shared DB Size	250GB

Test Scenarios

The tests modeled a typical web commerce scenario of anonymous users browsing the site and purchasing products. There were four main traffic scenarios: Browse Home, Browse Category, Browse Products, and Checkout.

Scenario	Request
Browse Home	• Visit Home Page
Browse Category	• Visit a random Category page
Browse Product	• Visit a random Product page • Visit another random Product page
Checkout	• Visit Home page • Visit a random Category page • Visit a random Product page • Add product to the cart • Complete checkout

Visit Home Page

Request: *{{SiteName}}*

Dependent Requests:

{{SiteName}}/layouts/System/VisitorIdentificationCSS.aspx

{{SiteName}}/api/cxa/Cart/GetCartLinesCount

{{SiteName}}/api/cxa/Catalog/GetPromotedProducts (x2)

© Copyright 2019, Sitecore - all rights reserved.

Visit Category Page

Request: *{{SiteName}}/category/{CategoryName}}*

Dependent Requests:

{{SiteName}}/layouts/System/VisitorIdentificationCSS.aspx

{{SiteName}}/api/cxa/Catalog/GetProductList

Visit Product Page

Request: *{{SiteName}}/product/{ProductId}}*

Dependent Requests:

{{SiteName}}/layouts/System/VisitorIdentificationCSS.aspx

{{SiteName}}/api/cxa/Catalog/GetCurrentProductStockInfo

Add To Cart

Request: *{{SiteName}}/api/cxa/Cart/AddCartLine*

Dependent Requests:

{{SiteName}}/api/cxa/Cart/GetCartLinesCount

Checkout

Request: {{SiteName}}/checkout/delivery

Dependent Requests:

{{SiteName}}/layouts/System/VisitorIdentificationCSS.aspx

{{SiteName}}/api/cxa/checkout/GetDeliveryData

{{SiteName}}/api/cxa/Cart/GetCart

{{SiteName}}/api/cxa/checkout/GetShippingMethods

{{SiteName}}/api/cxa/checkout/SetShippingMethods

Request: {{SiteName}}/checkout/billing

Dependent Requests:

{{SiteName}}/api/cxa/checkout/GetBillingData

{{SiteName}}/api/cxa/Cart/GetCart

© Copyright 2019, Sitecore - all rights reserved.

Request: *{{SiteName}}/api/cxa/checkout/SetPaymentMethods*

Dependent Requests:

None

Request: *{{SiteName}}/checkout/review*

Dependent Requests:

{{SiteName}}/api/cxa/checkout/GetReviewData

{{SiteName}}/api/cxa/Cart/GetCart

Request: *{{SiteName}}/api/cxa/checkout/SubmitOrder*

Dependent Requests:

None

Load Matrix

Load Traffic Types

A warm-up period of 5 minutes followed by a gradual increase in users until the target of 70 is attained.

Load Matrix

Scenario	Percentage
Home	50%
Category	15%
Product	30%
Checkout	5%

Load Specifications

Configuration	Values
Warm-Up	5 minutes
Test Duration	4 hours
Cool-Down	N/A

User Specifications

© Copyright 2019, Sitecore - all rights reserved.

Configuration	Values
Max number of users	70
Think Time	2 seconds
Interaction Groups	Constant Load Pattern • Percentage New Users: 100 • Think Profile: Normal Distribution

Data Configuration

Catalog Data

- Products: 100K with 50% Relationships
- Categories: 1K
- Category Depth: 4
- Products/Category: 100

User Profiles

- 500K Users (with addresses)

Test Infrastructure Configuration

To generate the performance load, this test used a Visual Studio 2017 Load Test Rig with the following configuration:

- The Visual Studio 2017 test agents and the controller are deployed to Azure Large DSv3 instances.
- Each agent had a single data disk attached, and the agent service was set to utilize the attached disk as the working directory ensuring that the agent had enough disk capacity to handle load generation.

Performance Metrics

Metric Name	Source	Description
User Load	VS/Test Agent	The current number of users.
Requests/Sec	VS/Test Agent	Rate of all requests to the application per second, measured by the Test Agent.
Pages/Sec	VS/Test Agent	Rate of all pages (composed of a group of requests) per second, measured by the Test Agent.

© Copyright 2019, Sitecore - all rights reserved.

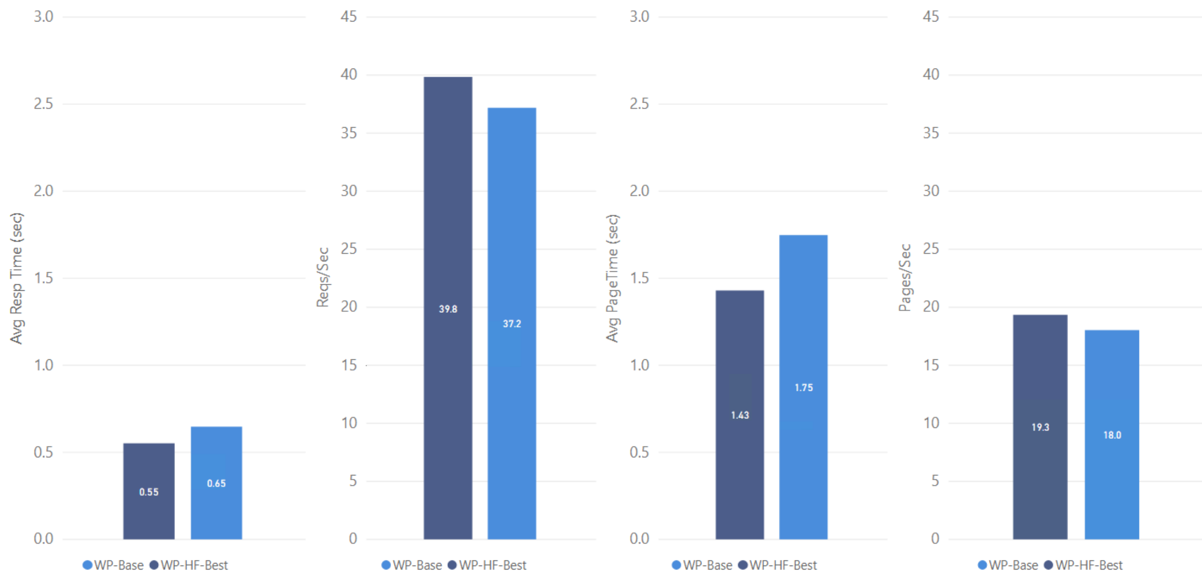
Metric Name	Source	Description
Response Time	VS/Test Agent, Azure	The time a request takes to respond to the Test Agent.
Page Time	VS/Test Agent	The time a page takes to respond to the Test Agent.
App Service Plan CPU %	Azure	The percentage of processing resources used.
App Service Memory Working Set	Azure	The set of memory pages or areas of memory allocated to an app service that was recently used by the threads in the app service.
DTU Consumption %	Azure	The percentage of CPU, memory, reads, and writes of a database, as a blended resource.

© Copyright 2019, Sitecore - all rights reserved.

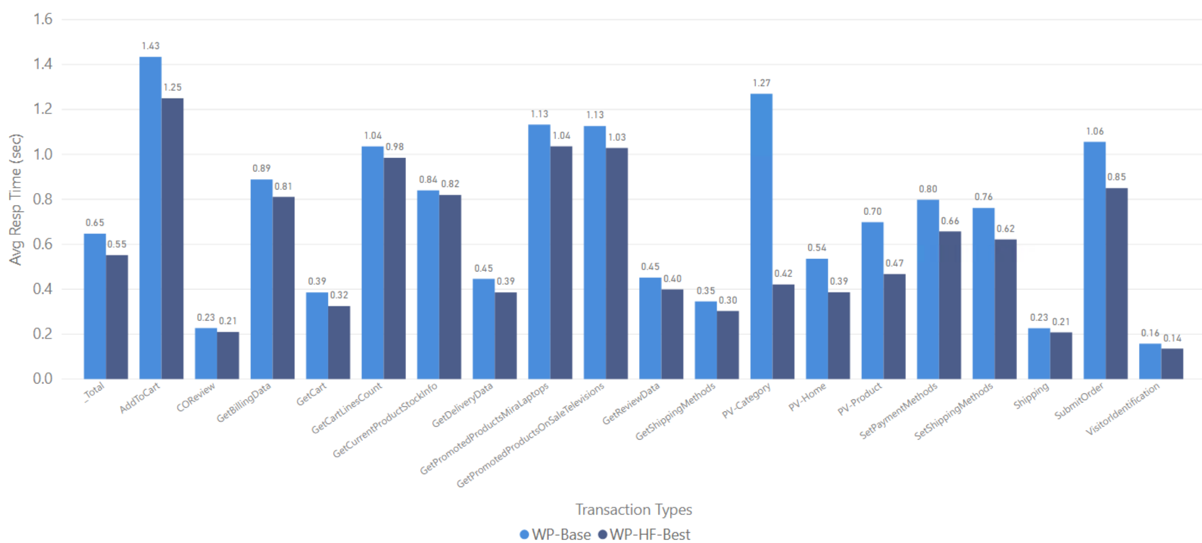
Results

Key Metrics

Test Summary



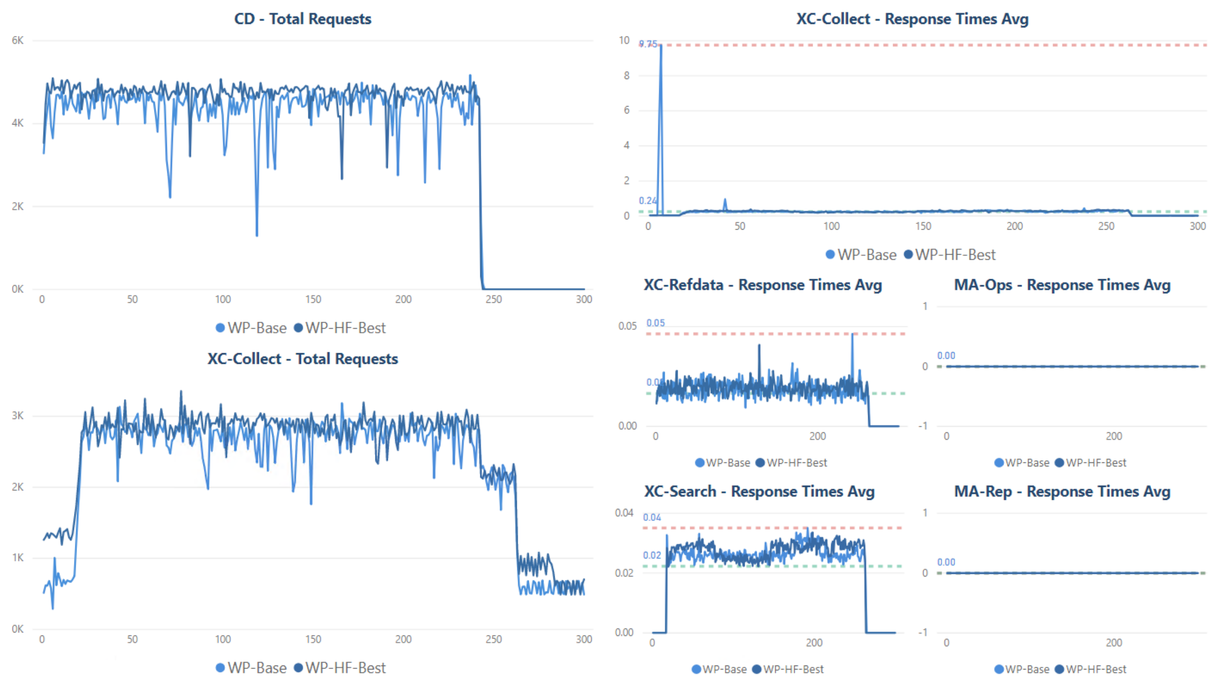
Avg Response Time by Transaction



© Copyright 2019, Sitecore - all rights reserved.

Response Times – CD, CM, PRC, REP, Shop, Auth, Minions

Response Times – XC-Collect, XC-Refdata, XC-Search, MA-Ops, MA-Rep



© Copyright 2019, Sitecore - all rights reserved.

CPU – App Service Plan

CD, CM, Engine, Minions

PRC, REP, XC-Resource Intensive, XC-Basic

© Copyright 2019, Sitecore - all rights reserved.

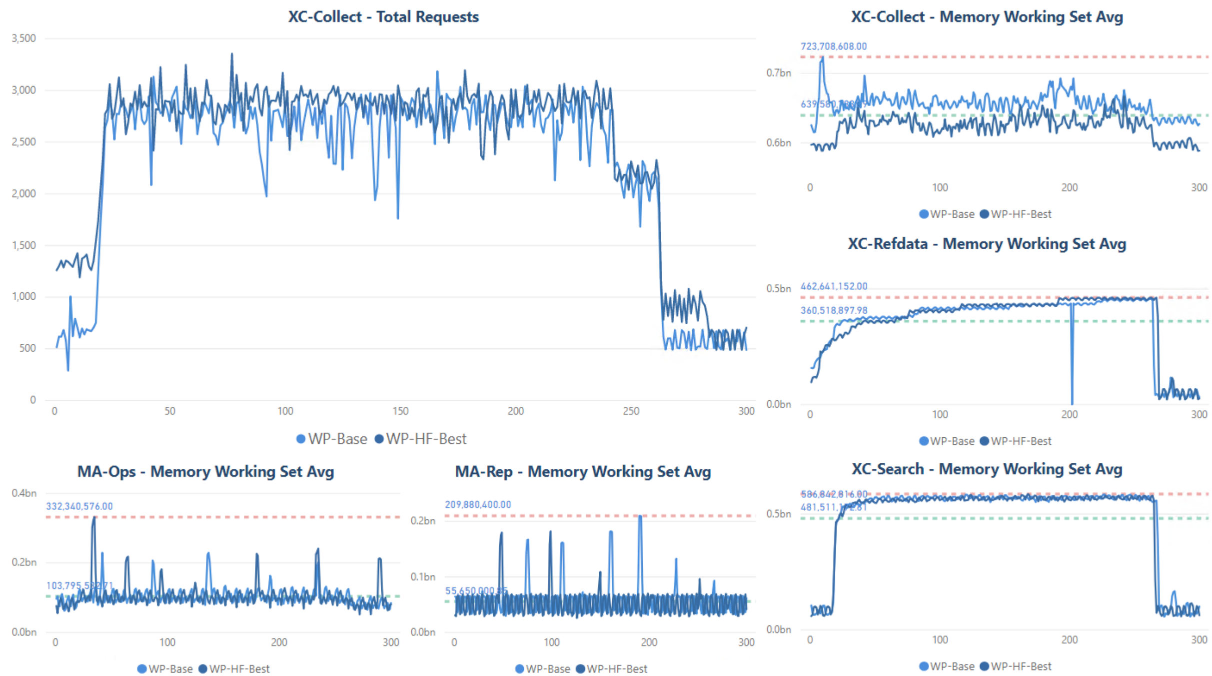
App Memory

CD, CM, Shop

Auth, Minions, PRC, REP

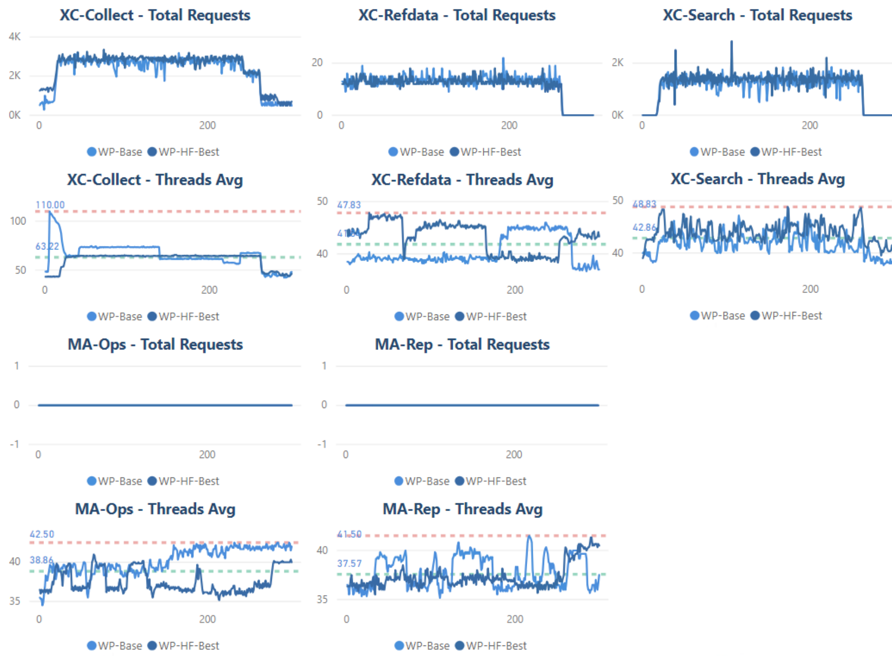
© Copyright 2019, Sitecore - all rights reserved.

XC-Collect, XC-Refdata, XC-Search, MA-Ops, MA-Rep

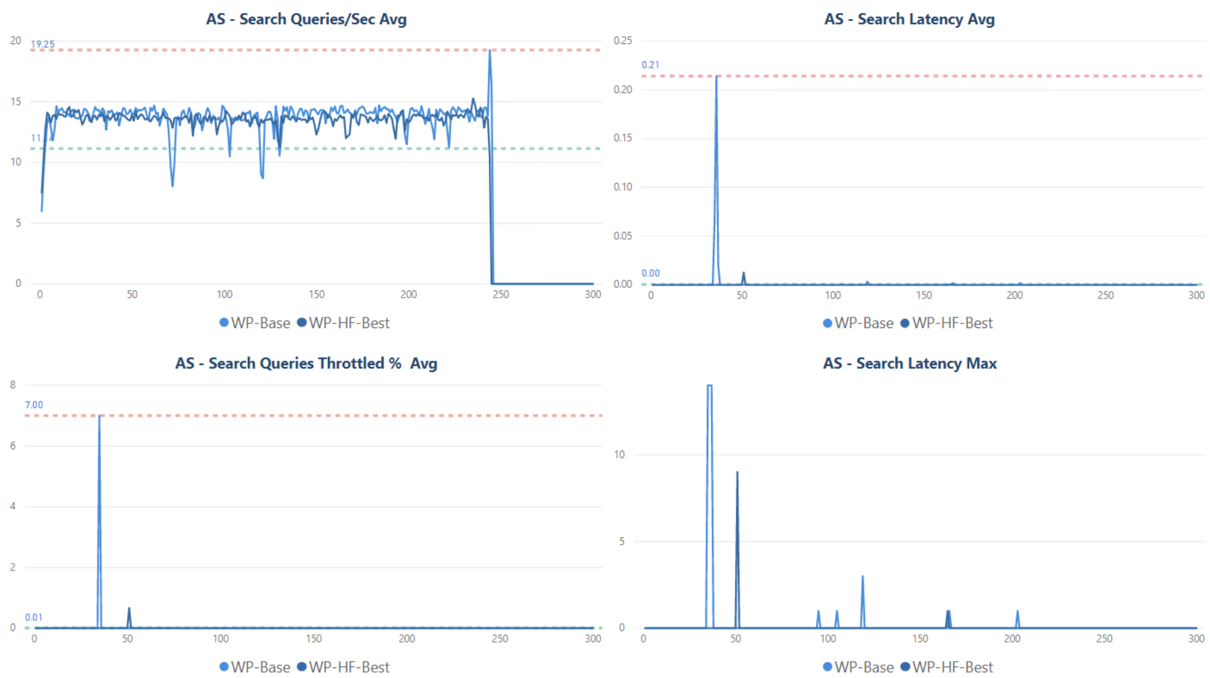


Threads

© Copyright 2019, Sitecore - all rights reserved.



Azure Search



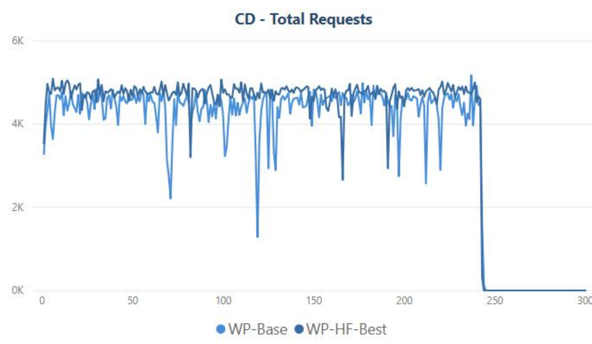
© Copyright 2019, Sitecore - all rights reserved.

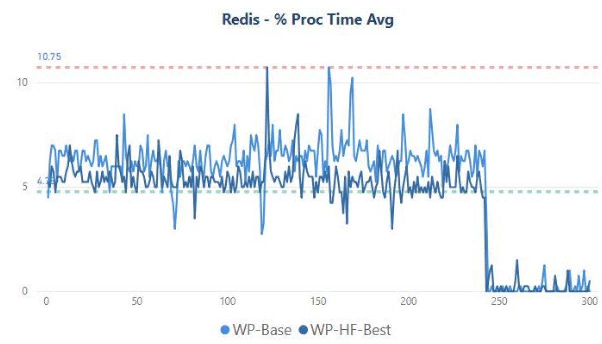
Azure SQL Databases

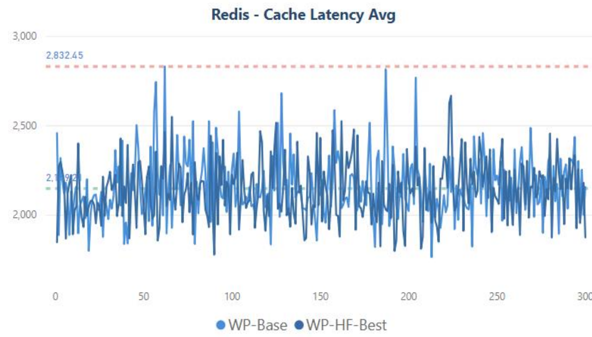
Azure Cache – Redis

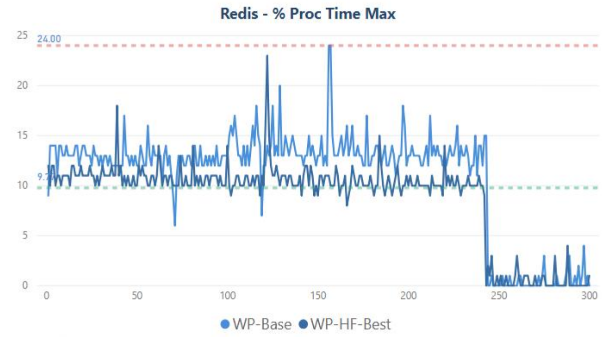
© Copyright 2019, Sitecore - all rights reserved.

Azure Cache – Commerce Redis









© Copyright 2019, Sitecore - all rights reserved.

Appendix

Thread Starvation Hotfix

This hotfix addresses the thread starvation issue in the scenario detailed below.

When running consistent/stable traffic (that is, static number of users performing site browsing and checkouts) for long durations (12hrs), Redis gets into a situation where the max memory and # of keys steadily increases until Redis runs out of memory or the test/traffic stops (**Fig TSH-1**). Note that the entire deployment is monitored and that there are no resources reaching any critical thresholds, whether it be CPU, Memory, and so on. What is especially odd is that the system appears to be in a stable state, and then something that is, as yet, unknown happens. After that, Redis memory/keys start climbing.

The rise coincides with a steady increase in the number of threads on the CDs (**Fig TSH-2**). Also, an examination of memory dumps taken at different intervals shows that there are numerous "/SESSION END" requests that are still being processed after 500+ seconds (**Fig TSH-3**), as well as unknown requests (that may or may not be XConnect related) that have lingered over 8000s (**Fig TSH-4a**). Looking at the associated items on the HTTP context of the unknown request, it shows data related to "ContactLockIds" (**Fig TSH-4b**). Unfortunately, there are no threads associated with the unknown requests.

Similar behavior was encountered in the field (using 9.0.2 of Sitecore, 9.0.3 of XC). Note that the deployments are not the exactly the same (customer was on-premise and was using a Redis Cluster), but the behaviors can still be relevant.

This situation can happen without warning, even when all the resources look healthy. If the traffic is sustained long enough and the allocated Redis memory is exceeded, then the throughput in the system decreases dramatically, while the response times spike accordingly (**Fig TSH-5**). Sometimes, the system can recover after a certain period.

Fig TSH-1

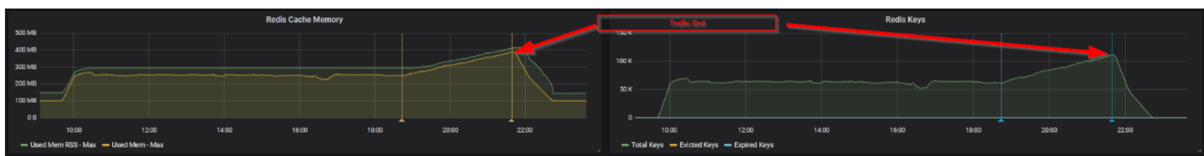
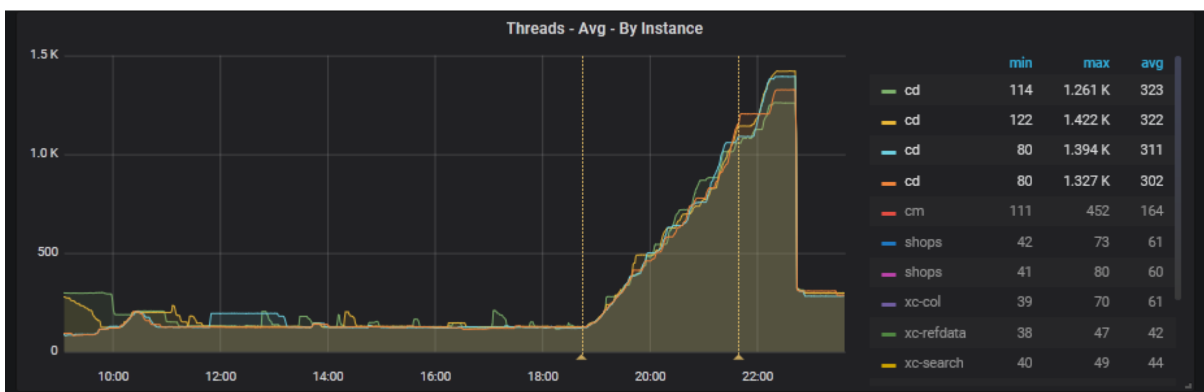


Fig TSH-2



© Copyright 2019, Sitecore - all rights reserved.

Fig TSH-3

```
AS 0000017511862f58     no    0     679        200
AS 000001751188d8f0     no    0     675        200      /SESSION END
AS 0000017511a0efa8     no    0     644        200      /SESSION END
AS 0000017511ae73f8     no    0     627        200      /SESSION END
AS 0000017511b36170     no    0     625        200      /SESSION END
AS 0000017511b9dba8     no    0     619        200
   0000017511c648f0    yes  600                200 POST /api/cxa/Catalog/GetPromotedProducts?sc_site=Storefront
AS 0000017511d6c7f8     no    0     589        200
   0000017511d760a8    yes  600                200 POST /api/cxa/Cart/GetCartLinesCount?sc_site=Storefront
AS 0000017511de35e0     no    0     579        200      /SESSION END
AS 0000017511e7ad88     no    0     569        200      /SESSION END
AS 0000017511ed09b0     no    0     559        200
AS 0000017511f33698     no    0     551        200      /SESSION END
AS 0000017511f802d0     no    0     547        200      /SESSION END
AS 00000175120d1f60     no    0     525        200      /SESSION END
AS 000001751211 1e30    no    0     523        200      /SESSION END
AS 00000175 1213d288    no    0     520        200      /SESSION END
AS 000001751216dd30     no    0     520        200      /SESSION END
```

Fig TSH-4a

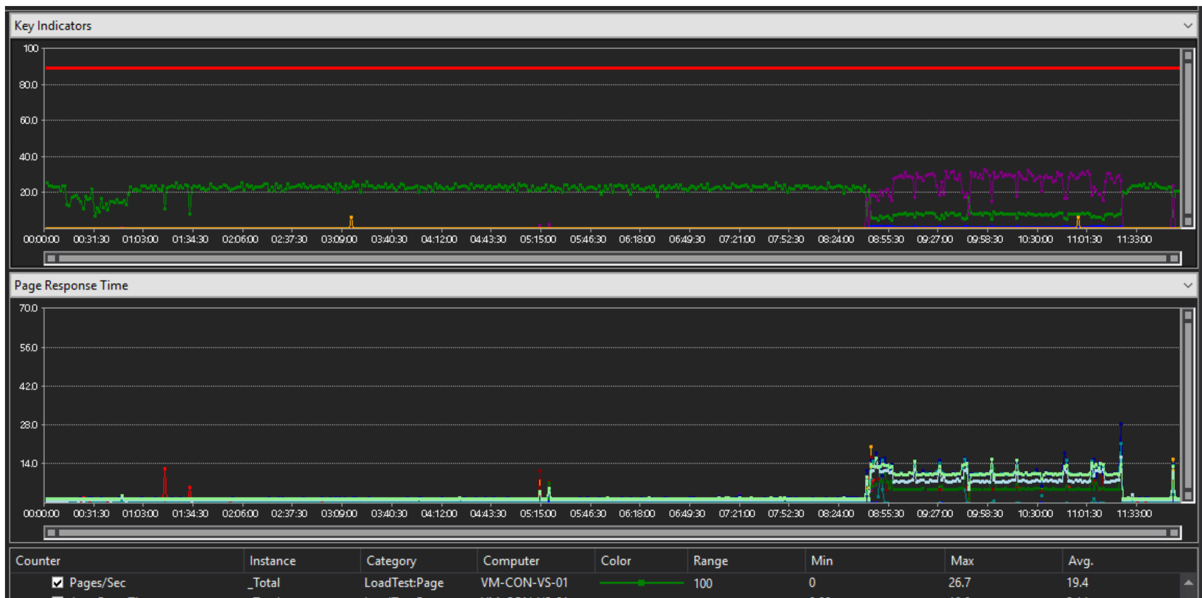
```
   Address          Completed Timeout Time (secs) ThreadId ReturnCode Verb Url
AS 0000017510d9a9d8        no       0        8391                 200
AS 0000017510dd0740        no       0        8271                 200
AS 0000017510df00b8        no       0        8091                 200
AS 0000017510e06db0        no       0        7941                 200
AS 0000017510e1d0b0        no       0        7791                 200
AS 0000017510e4e028        no       0        7521                 200
AS 0000017510e54700        no       0        7491                 200
AS 0000017510e6e770        no       0        7311                 200
AS 0000017510e74040        no       0        7281                 200
AS 0000017510e84a10        no       0        7191                 200
AS 0000017510ea3b50        no       0        7011                 200
AS 0000017510eb2fa0        no       0        6921                 200
AS 0000017510ebd058        no       0        6861                 200
AS 0000017510ed2260        no       0        6741                 200
AS 0000017510edde00        no       0        6681                 200
AS 0000017510f0c220        no       0        6411                 200
AS 0000017510f17198        no       0        6351                 200
AS 0000017510f1c450        no       0        6321                 200
AS 0000017510f2f578        no       0        6170                 200
AS 0000017510f60cb8        no       0        5870                 200
AS 0000017510fad288        no       0        5420                 200
AS 0000017510fb4da0        no       0        5390                 200
AS 0000017510fbb0f8        no       0        5360                 200
AS 0000017510fec198        no       0        5120                 200
AS 000001751100c9b0        no       0        4940                 200
AS 000001751101de48        no       0        4850                 200
AS 0000017511024328        no       0        4820                 200
```

Fig TSH-4b

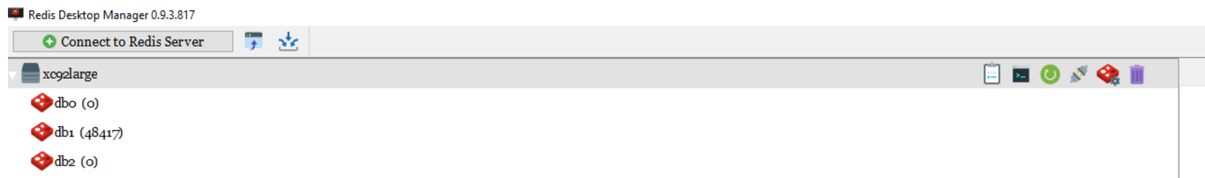
```
[raw] 0000017510d9b090 System.Collections.Hashtable Entries: 1

Key:   00000178113ea3e0  "ContactLockIds" [14] (System.String)
Value: 0000017510d9b140 (System.Collections.Generic.Dictionary<System.Guid,System.Object>)
```


© Copyright 2019, Sitecore - all rights reserved.

Fig TSH-5

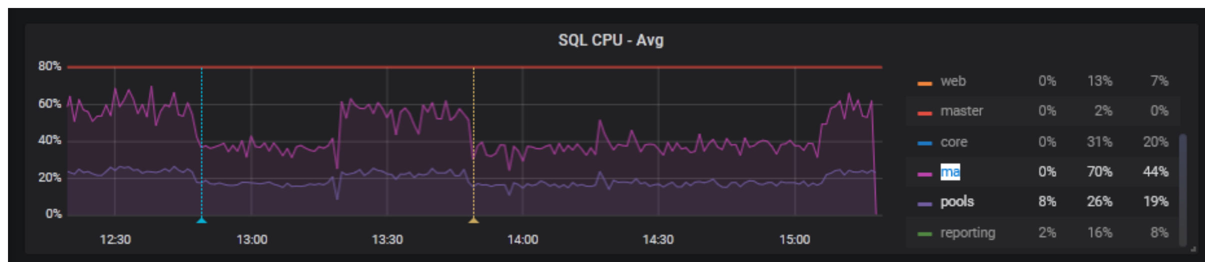


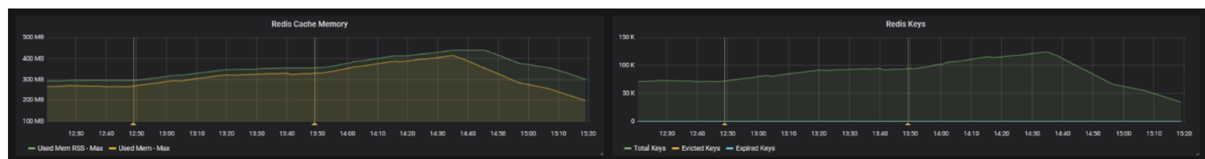
Private Session Processing



45m after the end of the run @ 34333 keys (private)

1h14m after the end of the run @ 0 keys





© Copyright 2019, Sitecore - all rights reserved.

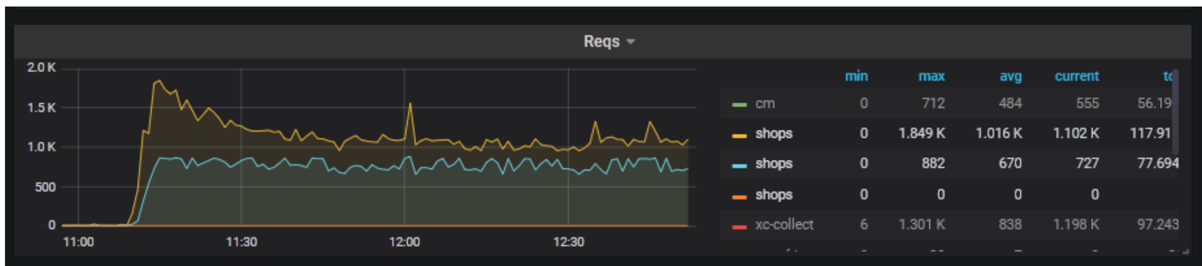
Setting a polling interval of 30 for private sessions seems to prevent the situation where there are "stuck" keys.

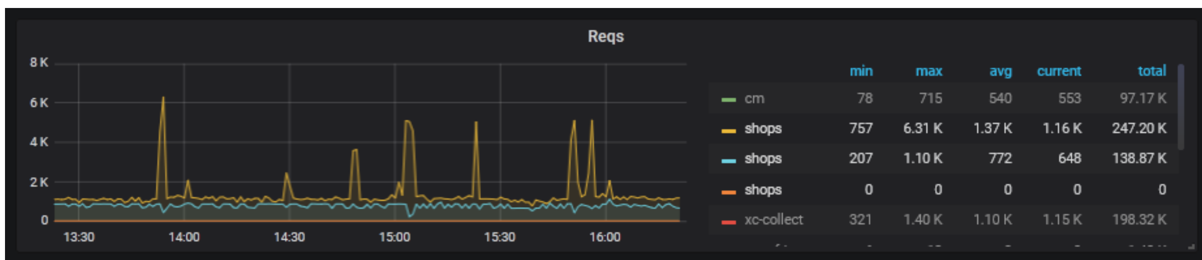
- Play around with pollingMaxExpiredSessionsPerSecond to increase processing output of Private Sessions
    - pollingInterval=2, pollingMaxExpiredSessionsPerSecond=10 - Baseline
    - pollingInterval=15, pollingMaxExpiredSessionsPerSecond=10 - Higher MA processing; no stuck private sessions, but lower response times when compared to baseline
    - pollingInterval=30, pollingMaxExpiredSessionsPerSecond=2010 - Higher MA processing; no stuck private sessions, but lower response times when compared to baseline

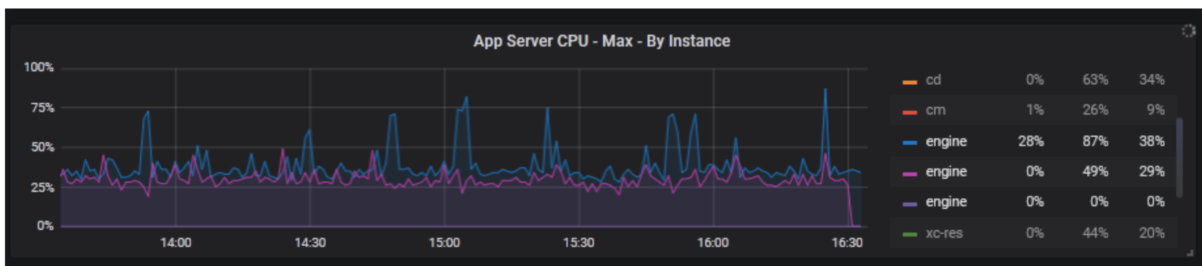
Shops ARR Affinity

ARR Affinity=On:

The following supplemental graphs support the key findings described in the previous section:

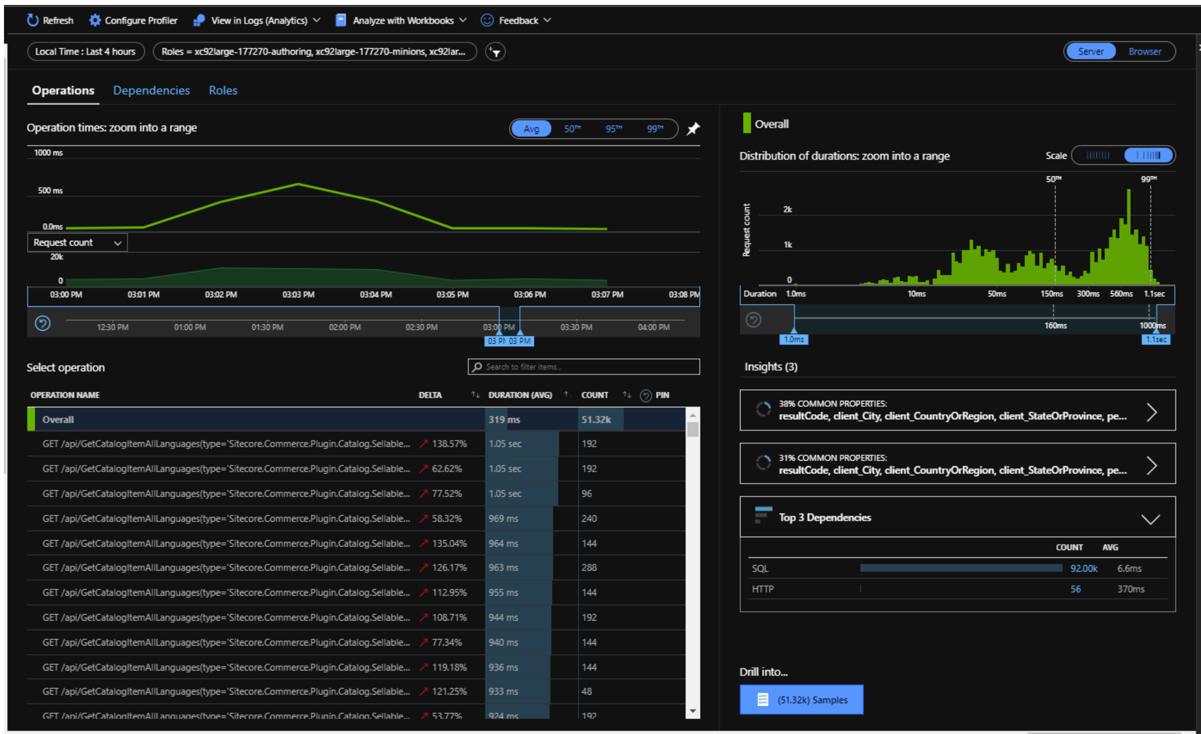






© Copyright 2019, Sitecore - all rights reserved.

The traffic spikes are primarily caused by the call to "/api/GetCatalogItemAllLanguages type=Sitecore.Commerce.Plugin.Catalog.SellableItem".



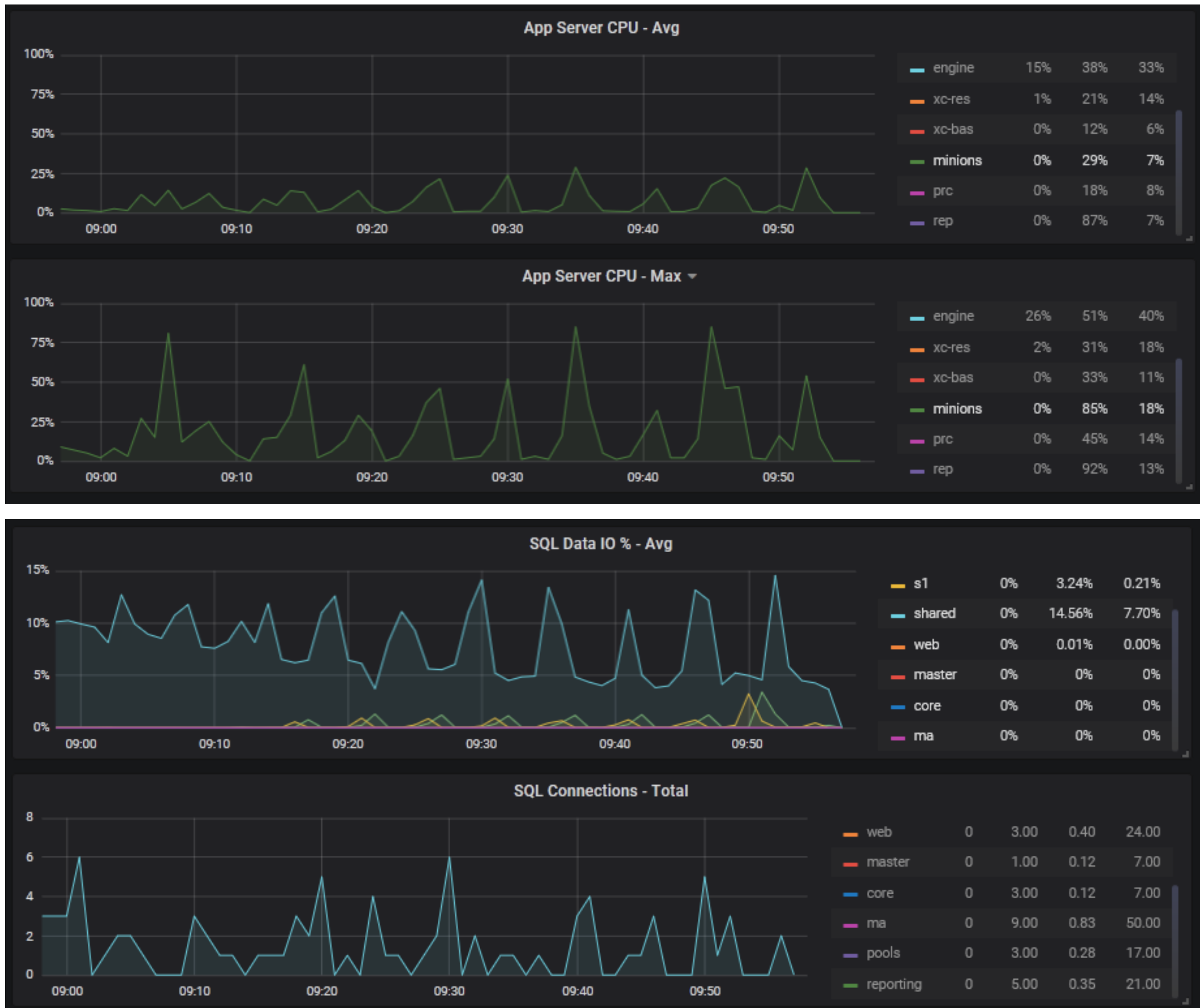
ARR Affinity=Off:

The following supplemental graphs support the key findings described in the previous section:

© Copyright 2019, Sitecore - all rights reserved.

Minions Wake-Up Interval

The following supplemental graphs support the key findings described in the previous section:



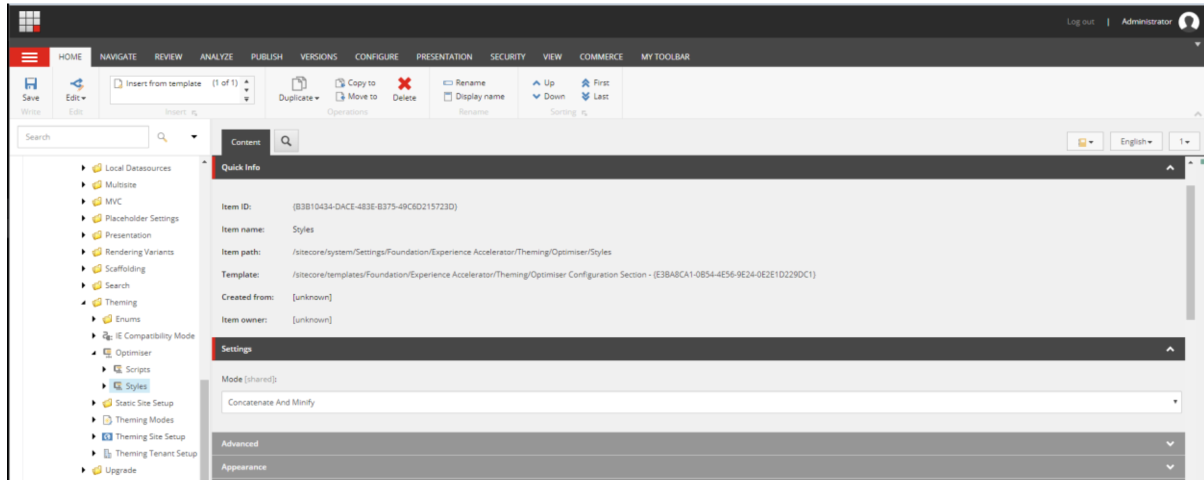
© Copyright 2019, Sitecore - all rights reserved.

Experience Accelerator Theming

The following setting is recommended to improve the general performance of every page on the site:

In the Content Editor, in the **Mode** field, set the following to **Concatenate and Minify**:

- *# /sitecore/system/Settings/Foundation/Experience Accelerator/Theming/Optimiser/Scripts*
- *# /sitecore/system/Settings/Foundation/Experience Accelerator/Theming/Optimiser/Styles*



© Copyright 2019, Sitecore - all rights reserved.