Performance White Paper

Sitecore Email Experience Manager 10.0

Sitecore EXM 10.0 Performance Testing

Table of Contents

Executive summary

This white paper describes Sitecore Email Experience Manager (EXM) performance testing.

EXM is one of the main parts of the Sitecore Experience Platform. It allows you to create individual email campaigns and make them both personal and relevant for all clients.

The objectives of EXM performance testing are to:

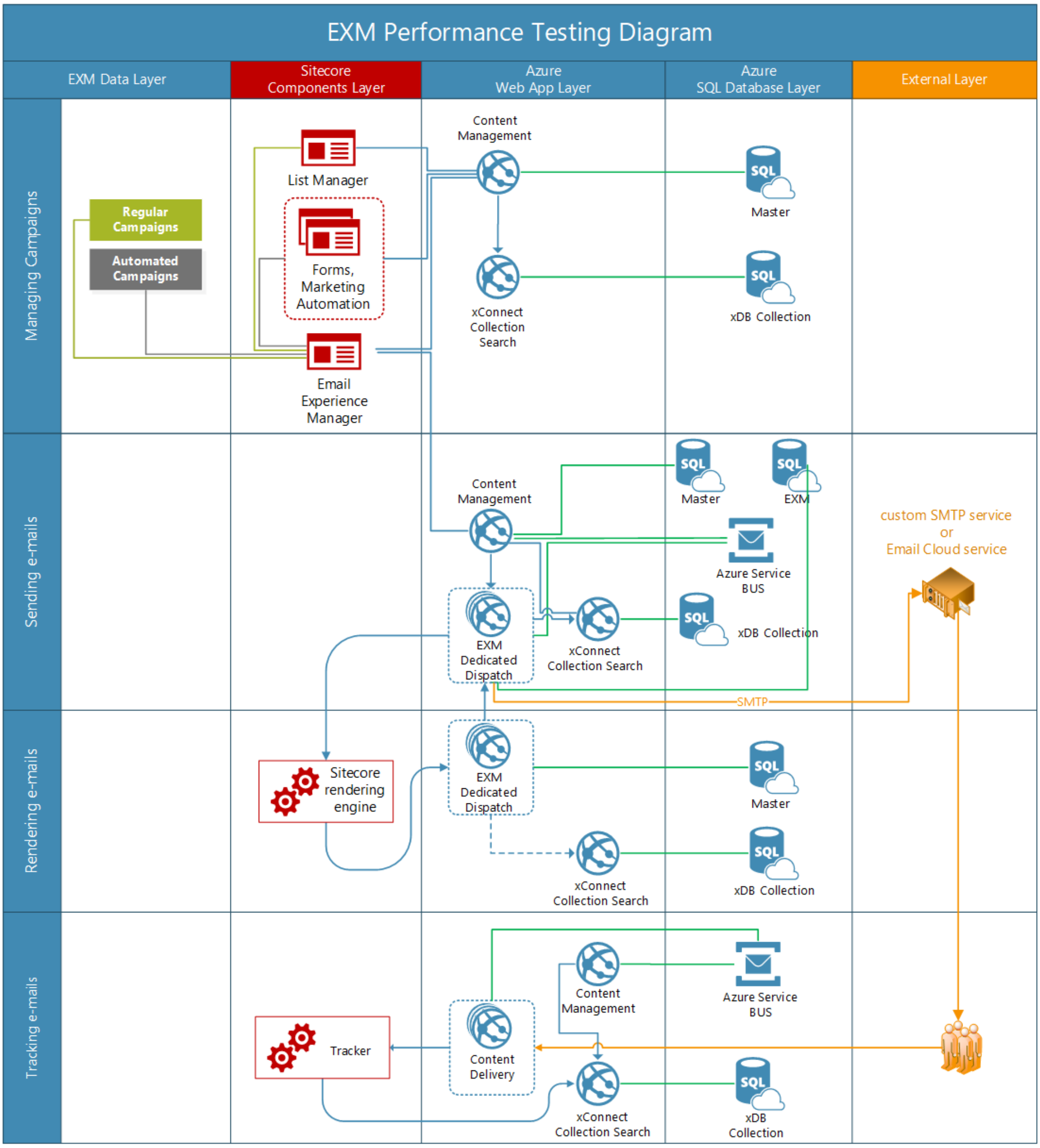
- Check the ability of the system to operate at the expected load levels.
- Establish a baseline performance based on a “large” deployment.
- Identify possible bottlenecks in the system.

The tests focus on the following three main domains:

- Content Management
- Dedicated EXM server
- Content Delivery

EXM overview

EXM is an integrated part of the Sitecore Experience Platform that relies heavily on the tracking, reporting, and segmentation functionality of the Sitecore Experience Database (xDB). If you use Sitecore Experience Manager, you can use the email campaign capabilities to manage and personalize emails, and deliver personalized messages to your customers.



Managing email campaigns

You manage email campaigns in the EXM application that is served by the Content Management role. The campaigns are stored in the Master database.

In this white paper, we focus on two types of campaign:

- Regular email campaigns – email campaigns with no predefined content or recipients to create a new email campaign from scratch.
- Personalized email campaigns – used for sending specific messages to targeted audiences and ensuring that the right content is sent to the right audience. That is, campaigns that only displays content that each recipient has shown an interest in, content based on accumulated profile values, or content based on the pattern card that matches the recipient the best.

Email campaigns can be associated with contact lists or segments that you manage through the List Manager. In the List Manager, there are two types of lists: contact lists and segmented lists. A contact list targets a specific group of contacts. A segmented list filters an existing contact list. Both types of lists are stored in the xDB. The List Manager API uses the xConnect Collection Search role to retrieve contacts in lists.

You can associate an email campaign with multiple contact lists or segmented lists. You can also use lists to select the contacts to include or exclude in the dispatch process.

Sending emails

The EXM Dispatch role sends a specific email message to a contact. Depending on the configuration, you can use a custom SMTP service or the Email Cloud service using SMTP to send the message.

The EXM Dispatch role then saves an Email Sent interaction to the xConnect Collection Search role. This records the email campaign on the individual contact in the xDB, provides reporting in EXM and Experience Analytics, and enables personalization across other channels.

The EXM Dispatch role then removes the contact from the dispatch queue in the EXM database.

During the dispatch job, the Content Management role waits for job completion on all EXM Dispatch roles. When all EXM Dispatch roles are done and all contacts in the EXM database queue have been processed, the Content Management role changes the email campaign's state in the Master database to *Sent*.

Rendering emails

The EXM Dispatch role renders an email message by sending an HTTP request and generates a page through the Sitecore rendering engine. The email message contains content items that come from the Master database.

For simple, personalized email messages that use basic token replacement, the EXM Dispatch role generates the email message once and caches it for all contacts. For highly personalized emails, an HTTP request is sent to the EXM Dispatch role, and email messages render separately for each contact. Therefore, personalization puts a significantly higher load on the EXM Dispatch role, which makes scaling considerations important.

For reporting purposes, the EXM Dispatch role saves a message on the Message Bus to update the email address history facet.

The Content Management role handles the message and updates the contact through the xConnect Collection Search role.

Tracking emails

After EXM sends an email message and the recipient opens it, the Content Delivery role receives a request. The request returns an empty 1x1 pixel.

The Content Delivery role handles the request and stores an *Open event* message in the Message Bus.

The Content Management role handles this message and stores an interaction for the contact that contains an Open event in xConnect Collection role. An EXM-specific xConnect plugin runs on the xConnect Collection Search role to update the specific contact facets on the contact that Sitecore uses for reporting purposes.

All the links in the email campaigns are associated with a specific EXM tracking page. When a recipient clicks an email link, the tracking page uses the standard tracker to create an interaction for the contact and registers a *Click page* event. The tracking page then redirects to the actual page in the link.

The tracker uses the regular tracking and collections data flows. When a session ends, the Content Delivery role sends the interaction to the xConnect Collection role. An EXM-specific plugin runs on the xConnect Collection role to update specific facets for reporting purposes.

EXM performance testing approach

The purpose of testing is to make sure that Sitecore 10.0 EXM can handle 5 million messages per month and identify possible bottlenecks in the system.

In consideration of the business logic and features of working with email companies, the EXM performance testing is divided into the following phases:

Phase I. Sending emails

In this phase, we have combined the following processes:

- Managing email campaigns
- Rendering emails
- Sending emails

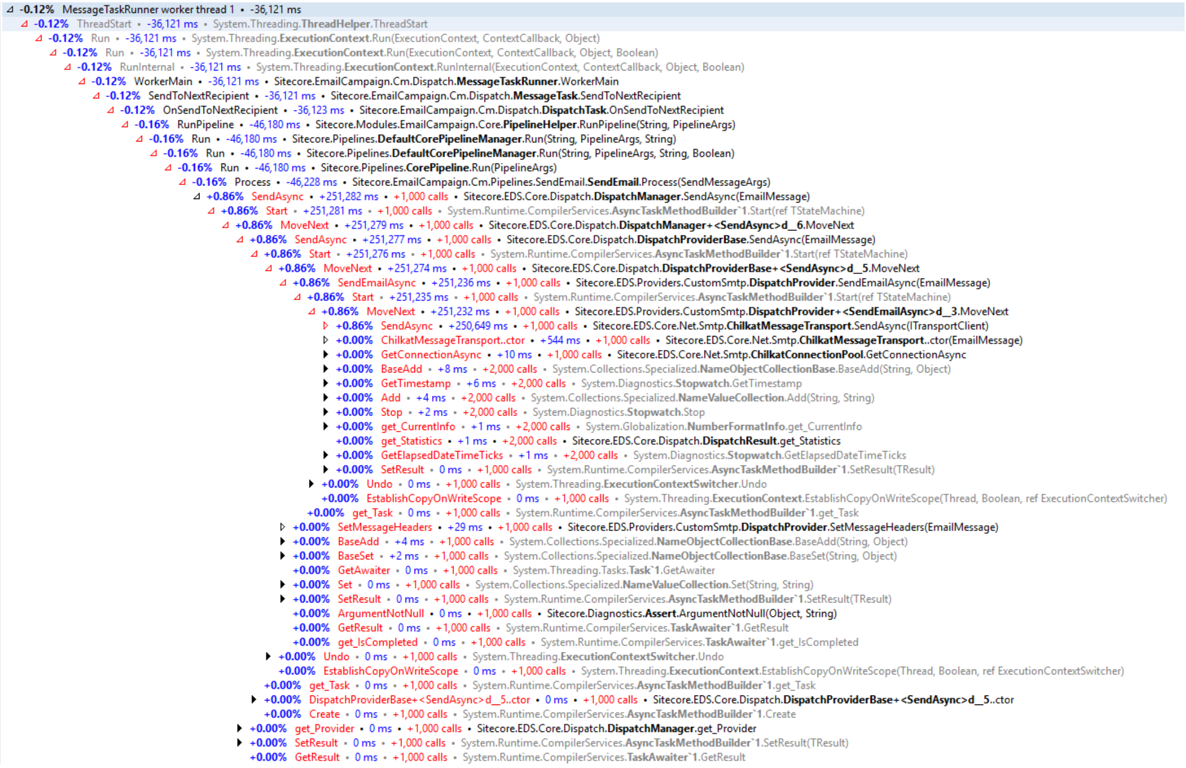
The sending emails process uses third-party solutions to send email messages (SMPT server, Email Cloud services), and we therefore decided to use a stub to bypass third-party applications.

EXM lets you test campaign throughput by emulating a message transfer agent (MTA). MTA emulation lets you imitate the round-trip time required to send an email message from the Sitecore CMS to the MTA. For more information, see the article Testing EXM performance in emulation mode.

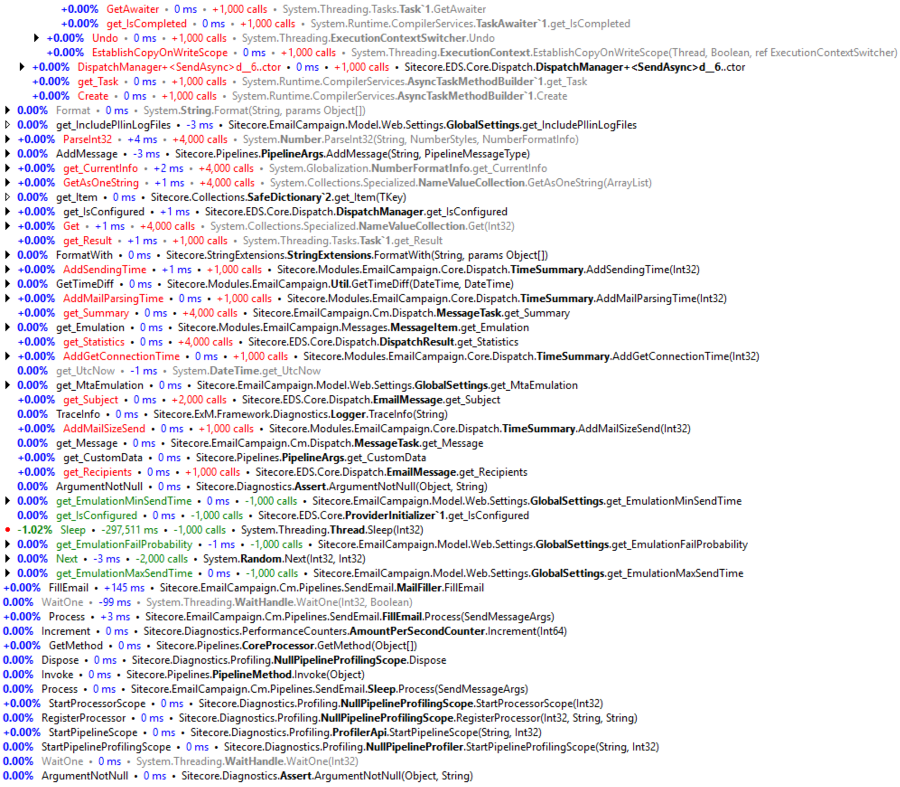
Unfortunately, this solution has some limitations:

- Emulation sending mode does not move the message to the *Sent* state – the message status changes back to *Drafts* after the process has been completed.
- Emulation sending mode is not available for message variants when you run an A/B test.
- The most critical limitation for performance testing is that most of the email sending pipeline (such as marketing automation, processing, and reporting pipelines) remains unused when you use this solution.

The unused methods are highlighted in red in the following screenshots:



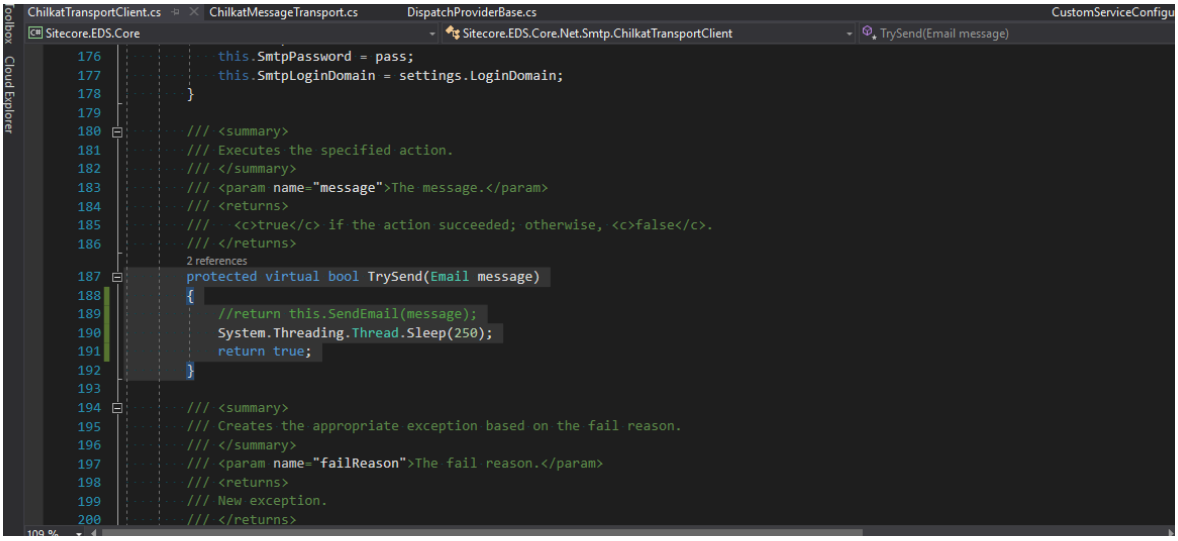
```
⊿ -0.12%  MessageTaskRunner worker thread 1 • -36,121 ms
  ⊿ -0.12%  ThreadStart • -36,121 ms • System.Threading.ThreadHelper.ThreadStart
    ⊿ -0.12%  Run • -36,121 ms • System.Threading.ExecutionContext.Run(ExecutionContext, ContextCallback, Object)
      ⊿ -0.12%  Run • -36,121 ms • System.Threading.ExecutionContext.Run(ExecutionContext, ContextCallback, Object, Boolean)
        ⊿ -0.12%  RunInternal • -36,121 ms • System.Threading.ExecutionContext.RunInternal(ExecutionContext, ContextCallback, Object, Boolean)
          ⊿ -0.12%  WorkerMain • -36,121 ms • Sitecore.EmailCampaign.Cm.Dispatch.MessageTaskRunner.WorkerMain
            ⊿ -0.12%  SendToNextRecipient • -36,121 ms • Sitecore.EmailCampaign.Cm.Dispatch.MessageTask.SendToNextRecipient
              ⊿ -0.12%  OnSendToNextRecipient • -36,123 ms • Sitecore.EmailCampaign.Cm.Dispatch.DispatchTask.OnSendToNextRecipient
                ⊿ -0.16%  RunPipeline • -46,180 ms • Sitecore.Modules.EmailCampaign.Core.PipelineHelper.RunPipeline(String, PipelineArgs)
                  ⊿ -0.16%  Run • -46,180 ms • Sitecore.Pipelines.DefaultCorePipelineManager.Run(String, PipelineArgs, String)
                    ⊿ -0.16%  Run • -46,180 ms • Sitecore.Pipelines.DefaultCorePipelineManager.Run(String, PipelineArgs, String, Boolean)
                      ⊿ -0.16%  Run • -46,180 ms • Sitecore.Pipelines.CorePipeline.Run(PipelineArgs)
                        ⊿ -0.16%  Process • -46,228 ms • Sitecore.EmailCampaign.Cm.Pipelines.SendEmail.SendEmail.Process(SendMessageArgs)
                          ⊿ +0.86%  SendAsync • +251,282 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.DispatchManager.SendAsync(EmailMessage)
                            ⊿ +0.86%  Start • +251,281 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.Start(ref TStateMachine)
                              ⊿ +0.86%  MoveNext • +251,279 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.DispatchManager+<SendAsync>d__6.MoveNext
                                ⊿ +0.86%  SendAsync • +251,277 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.DispatchProviderBase.SendAsync(EmailMessage)
                                  ⊿ +0.86%  Start • +251,276 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.Start(ref TStateMachine)
                                    ⊿ +0.86%  MoveNext • +251,274 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.DispatchProviderBase+<SendAsync>d__5.MoveNext
                                      ⊿ +0.86%  SendEmailAsync • +251,236 ms • +1,000 calls • Sitecore.EDS.Providers.CustomSmtp.DispatchProvider.SendEmailAsync(EmailMessage)
                                        ⊿ +0.86%  Start • +251,235 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.Start(ref TStateMachine)
                                          ⊿ +0.86%  MoveNext • +251,232 ms • +1,000 calls • Sitecore.EDS.Providers.CustomSmtp.DispatchProvider+<SendEmailAsync>d__3.MoveNext
                                            ▷ +0.86%  SendAsync • +250,649 ms • +1,000 calls • Sitecore.EDS.Core.Net.Smtp.ChilkatMessageTransport.SendAsync(ITransportClient)
                                            ▷ +0.00%  ChilkatMessageTransport..ctor • +544 ms • +1,000 calls • Sitecore.EDS.Core.Net.Smtp.ChilkatMessageTransport..ctor(EmailMessage)
                                            ▶ +0.00%  GetConnectionAsync • +10 ms • +1,000 calls • Sitecore.EDS.Core.Net.Smtp.ChilkatConnectionPool.GetConnectionAsync
                                            ▶ +0.00%  BaseAdd • +8 ms • +2,000 calls • System.Collections.Specialized.NameObjectCollectionBase.BaseAdd(String, Object)
                                            ▶ +0.00%  GetTimestamp • +6 ms • +2,000 calls • System.Diagnostics.Stopwatch.GetTimestamp
                                            ▶ +0.00%  Add • +4 ms • +2,000 calls • System.Collections.Specialized.NameValueCollection.Add(String, String)
                                            ▶ +0.00%  Stop • +2 ms • +2,000 calls • System.Diagnostics.Stopwatch.Stop
                                            ▶ +0.00%  get_CurrentInfo • +1 ms • +2,000 calls • System.Globalization.NumberFormatInfo.get_CurrentInfo
                                              +0.00%  get_Statistics • +1 ms • +2,000 calls • Sitecore.EDS.Core.Dispatch.DispatchResult.get_Statistics
                                            ▶ +0.00%  GetElapsedDateTimeTicks • +1 ms • +2,000 calls • System.Diagnostics.Stopwatch.GetElapsedDateTimeTicks
                                            ▶ +0.00%  SetResult • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.SetResult(TResult)
                                          ▶ +0.00%  Undo • 0 ms • +1,000 calls • System.Threading.ExecutionContextSwitcher.Undo
                                            +0.00%  EstablishCopyOnWriteScope • 0 ms • +1,000 calls • System.Threading.ExecutionContext.EstablishCopyOnWriteScope(Thread, Boolean, ref ExecutionContextSwitcher)
                                          +0.00%  get_Task • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.get_Task
                                      ▷ +0.00%  SetMessageHeaders • +29 ms • +1,000 calls • Sitecore.EDS.Providers.CustomSmtp.DispatchProvider.SetMessageHeaders(EmailMessage)
                                      ▶ +0.00%  BaseAdd • +4 ms • +1,000 calls • System.Collections.Specialized.NameObjectCollectionBase.BaseAdd(String, Object)
                                      ▶ +0.00%  BaseSet • +2 ms • +1,000 calls • System.Collections.Specialized.NameObjectCollectionBase.BaseSet(String, Object)
                                        +0.00%  GetAwaiter • 0 ms • +1,000 calls • System.Threading.Tasks.Task`1.GetAwaiter
                                      ▶ +0.00%  Set • 0 ms • +1,000 calls • System.Collections.Specialized.NameValueCollection.Set(String, String)
                                      ▶ +0.00%  SetResult • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.SetResult(TResult)
                                        +0.00%  ArgumentNotNull • 0 ms • +1,000 calls • Sitecore.Diagnostics.Assert.ArgumentNotNull(Object, String)
                                        +0.00%  GetResult • 0 ms • +1,000 calls • System.Runtime.CompilerServices.TaskAwaiter`1.GetResult
                                        +0.00%  get_IsCompleted • 0 ms • +1,000 calls • System.Runtime.CompilerServices.TaskAwaiter`1.get_IsCompleted
                                    ▶ +0.00%  Undo • 0 ms • +1,000 calls • System.Threading.ExecutionContextSwitcher.Undo
                                      +0.00%  EstablishCopyOnWriteScope • 0 ms • +1,000 calls • System.Threading.ExecutionContext.EstablishCopyOnWriteScope(Thread, Boolean, ref ExecutionContextSwitcher)
                                    +0.00%  get_Task • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.get_Task
                                  ▶ +0.00%  DispatchProviderBase+<SendAsync>d__5..ctor • 0 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.DispatchProviderBase+<SendAsync>d__5..ctor
                                    +0.00%  Create • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.Create
                                ▶ +0.00%  get_Provider • 0 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.DispatchManager.get_Provider
                                ▶ +0.00%  SetResult • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.SetResult(TResult)
                                  +0.00%  GetResult • 0 ms • +1,000 calls • System.Runtime.CompilerServices.TaskAwaiter`1.GetResult
```



```
          +0.00%  GetAwaiter • 0 ms • +1,000 calls • System.Threading.Tasks.Task`1.GetAwaiter
          +0.00%  get_IsCompleted • 0 ms • +1,000 calls • System.Runtime.CompilerServices.TaskAwaiter`1.get_IsCompleted
      ▶ +0.00%  Undo • 0 ms • +1,000 calls • System.Threading.ExecutionContextSwitcher.Undo
        +0.00%  EstablishCopyOnWriteScope • 0 ms • +1,000 calls • System.Threading.ExecutionContext.EstablishCopyOnWriteScope(Thread, Boolean, ref ExecutionContextSwitcher)
    ▶ +0.00%  DispatchManager+<SendAsync>d__6..ctor • 0 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.DispatchManager+<SendAsync>d__6..ctor
      +0.00%  get_Task • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.get_Task
      +0.00%  Create • 0 ms • +1,000 calls • System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1.Create
  ▶ 0.00%  Format • 0 ms • System.String.Format(String, params Object[])
  ▷ 0.00%  get_IncludePIlinLogFiles • -3 ms • Sitecore.EmailCampaign.Model.Web.Settings.GlobalSettings.get_IncludePIlinLogFiles
  ▶ +0.00%  ParseInt32 • +4 ms • +4,000 calls • System.Number.ParseInt32(String, NumberStyles, NumberFormatInfo)
  ▶ 0.00%  AddMessage • -3 ms • Sitecore.Pipelines.PipelineArgs.AddMessage(String, PipelineMessageType)
  ▶ +0.00%  get_CurrentInfo • +2 ms • +4,000 calls • System.Globalization.NumberFormatInfo.get_CurrentInfo
  ▶ +0.00%  GetAsOneString • +1 ms • +4,000 calls • System.Collections.Specialized.NameValueCollection.GetAsOneString(ArrayList)
  ▷ 0.00%  get_Item • 0 ms • Sitecore.Collections.SafeDictionary`2.get_Item(TKey)
  ▶ +0.00%  get_IsConfigured • +1 ms • Sitecore.EDS.Core.Dispatch.DispatchManager.get_IsConfigured
  ▶ +0.00%  Get • +1 ms • +4,000 calls • System.Collections.Specialized.NameValueCollection.Get(Int32)
    +0.00%  get_Result • +1 ms • +1,000 calls • System.Threading.Tasks.Task`1.get_Result
  ▶ 0.00%  FormatWith • 0 ms • Sitecore.StringExtensions.StringExtensions.FormatWith(String, params Object[])
  ▶ +0.00%  AddSendingTime • +1 ms • +1,000 calls • Sitecore.Modules.EmailCampaign.Core.Dispatch.TimeSummary.AddSendingTime(Int32)
  ▶ 0.00%  GetTimeDiff • 0 ms • Sitecore.Modules.EmailCampaign.Util.GetTimeDiff(DateTime, DateTime)
  ▶ +0.00%  AddMailParsingTime • 0 ms • +1,000 calls • Sitecore.Modules.EmailCampaign.Core.Dispatch.TimeSummary.AddMailParsingTime(Int32)
    +0.00%  get_Summary • 0 ms • +4,000 calls • Sitecore.EmailCampaign.Cm.Dispatch.MessageTask.get_Summary
  ▶ 0.00%  get_Emulation • 0 ms • Sitecore.Modules.EmailCampaign.Messages.MessageItem.get_Emulation
    +0.00%  get_Statistics • 0 ms • +4,000 calls • Sitecore.EDS.Core.Dispatch.DispatchResult.get_Statistics
  ▶ +0.00%  AddGetConnectionTime • 0 ms • +1,000 calls • Sitecore.Modules.EmailCampaign.Core.Dispatch.TimeSummary.AddGetConnectionTime(Int32)
    0.00%  get_UtcNow • -1 ms • System.DateTime.get_UtcNow
  ▶ 0.00%  get_MtaEmulation • 0 ms • Sitecore.EmailCampaign.Model.Web.Settings.GlobalSettings.get_MtaEmulation
    +0.00%  get_Subject • 0 ms • +2,000 calls • Sitecore.EDS.Core.Dispatch.EmailMessage.get_Subject
    0.00%  TraceInfo • 0 ms • Sitecore.ExM.Framework.Diagnostics.Logger.TraceInfo(String)
    +0.00%  AddMailSizeSend • 0 ms • +1,000 calls • Sitecore.Modules.EmailCampaign.Core.Dispatch.TimeSummary.AddMailSizeSend(Int32)
    0.00%  get_Message • 0 ms • Sitecore.EmailCampaign.Cm.Dispatch.MessageTask.get_Message
    +0.00%  get_CustomData • 0 ms • Sitecore.Pipelines.PipelineArgs.get_CustomData
    +0.00%  get_Recipients • 0 ms • +1,000 calls • Sitecore.EDS.Core.Dispatch.EmailMessage.get_Recipients
    0.00%  ArgumentNotNull • 0 ms • Sitecore.Diagnostics.Assert.ArgumentNotNull(Object, String)
  ▶ 0.00%  get_EmulationMinSendTime • 0 ms • -1,000 calls • Sitecore.EmailCampaign.Model.Web.Settings.GlobalSettings.get_EmulationMinSendTime
    0.00%  get_IsConfigured • 0 ms • -1,000 calls • Sitecore.EDS.Core.ProviderInitializer`1.get_IsConfigured
  ● -1.02%  Sleep • -297,511 ms • -1,000 calls • System.Threading.Thread.Sleep(Int32)
  ▶ 0.00%  get_EmulationFailProbability • -1 ms • -1,000 calls • Sitecore.EmailCampaign.Model.Web.Settings.GlobalSettings.get_EmulationFailProbability
  ▶ 0.00%  Next • -3 ms • -2,000 calls • System.Random.Next(Int32, Int32)
  ▶ 0.00%  get_EmulationMaxSendTime • 0 ms • -1,000 calls • Sitecore.EmailCampaign.Model.Web.Settings.GlobalSettings.get_EmulationMaxSendTime
▷ +0.00%  FillEmail • +145 ms • Sitecore.EmailCampaign.Cm.Pipelines.SendEmail.MailFiller.FillEmail
▷ 0.00%  WaitOne • -99 ms • System.Threading.WaitHandle.WaitOne(Int32, Boolean)
▶ +0.00%  Process • +3 ms • Sitecore.EmailCampaign.Cm.Pipelines.SendEmail.FillEmail.Process(SendMessageArgs)
▶ 0.00%  Increment • 0 ms • Sitecore.Diagnostics.PerformanceCounters.AmountPerSecondCounter.Increment(Int64)
▷ +0.00%  GetMethod • 0 ms • Sitecore.Pipelines.CoreProcessor.GetMethod(Object[])
  0.00%  Dispose • 0 ms • Sitecore.Diagnostics.Profiling.NullPipelineProfilingScope.Dispose
  0.00%  Invoke • 0 ms • Sitecore.Pipelines.PipelineMethod.Invoke(Object)
▶ 0.00%  Process • 0 ms • Sitecore.EmailCampaign.Cm.Pipelines.SendEmail.Sleep.Process(SendMessageArgs)
  +0.00%  StartProcessorScope • 0 ms • Sitecore.Diagnostics.Profiling.NullPipelineProfilingScope.StartProcessorScope(Int32)
  0.00%  RegisterProcessor • 0 ms • Sitecore.Diagnostics.Profiling.NullPipelineProfilingScope.RegisterProcessor(Int32, String, String)
▶ +0.00%  StartPipelineScope • 0 ms • Sitecore.Diagnostics.Profiling.ProfilerApi.StartPipelineScope(String, Int32)
  0.00%  StartPipelineProfilingScope • 0 ms • Sitecore.Diagnostics.Profiling.NullPipelineProfiler.StartPipelineProfilingScope(String, Int32)
  0.00%  WaitOne • 0 ms • System.Threading.WaitHandle.WaitOne(Int32)
  0.00%  ArgumentNotNull • 0 ms • Sitecore.Diagnostics.Assert.ArgumentNotNull(Object, String)
```

Obviously, to carry out a proper performance test of EXM, we needed to find another approach that meets performance testing requirements.

To do this, we changed the `TrySend()` method in the `ChilkatTransportClient` class that is located in the `Sitecore.EDS.Core.Net.Smtp:` namespace.



```
            this.SmtpPassword = pass;
            this.SmtpLoginDomain = settings.LoginDomain;
        }

        /// <summary>
        /// Executes the specified action.
        /// </summary>
        /// <param name="message">The message.</param>
        /// <returns>
        ///   <c>true</c> if the action succeeded; otherwise, <c>false</c>.
        /// </returns>
        protected virtual bool TrySend(Email message)
        {
            //return this.SendEmail(message);
            System.Threading.Thread.Sleep(250);
            return true;
        }

        /// <summary>
        /// Creates the appropriate exception based on the fail reason.
        /// </summary>
        /// <param name="failReason">The fail reason.</param>
        /// <returns>
        /// New exception.
        /// </returns>
```

Phase II. Tracking emails

All the links in the email campaigns are associated with a specific EXM tracking page. When a recipient clicks an email link, the tracking page uses the standard tracker to create an interaction for the contact and registers a *Click page* event. The tracking page then redirects to the actual page in the link.

The tracker uses the normal tracking and collections data flows. When a session ends, the Content Delivery role sends the interaction to the xConnect Collection role. An EXM-specific plugin runs on the xConnect Collection role to update specific facets for reporting purposes.

In the frame of this performance test, we tracked the following events:

- Open
- Click
- Unsubscribe
- Unsubscribe from all

Load profile

Load	Value	
Number of emails per month	5 million	
Number of emails per dispatch	100K	
Expected Hourly Send Rate	500K per hour	
Number of interactions with campaign	Open:	100%
	Click:	25%
	Unsubscribe:	3%
	Unsubscribe from List:	1%
	Unsubscribe from All:	1%
Email size	50 KB, 75 KB, 100 KB, 200 KB, 400 KB	

Deployment

Azure configuration

Azure topology	Sitecore topology configuration
XP “Large”	1 x CM 4 x CD (Open & Click Handling) 1 x xConnect Search 2 x xConnect Collection 2 x xConnect Ref 1 x Processing 1 x Reporting 1 x Dedicated Dispatch

Configuration and scaling

The EXM Dispatch server does not support Azure horizontal scaling, which is why we use 3x DDS servers.

Azure SQL Databases		App Service plans		
Name	**Pricing Tier**	**Role**	**Pricing Tier/Apps**	**Instances**
core-db	S1: 20 DTUs	CD-HP	S3: 1	4
exmmaster-db	S1: 20 DTUs	CM-HP	S3: 2	1
forms-db	S1: 20 DTUs	EXM-DDS-HP	S2: 1	1
ma-db	S1: 20 DTUs	PRC-HP	S2: 1	1
master-db	S1: 20 DTUs	REP-HP	*Combined with CM-HP*	
pools-db	S1: 20 DTUs	SI-HP	S2: 1	1
processingenginestorage-db	S3: 100 DTUs	XC-Basic-HP	S3: 4	1
processingenginetasks-db	S0: 10 DTUs	XC-ResourceIntensive-HP	S3: 3	2
refdata-db	S3: 100 DTUs			
reporting-db	S2: 50 DTUs			
shard0-db	P1: 125 DTUs			
shard1-db	P1: 125 DTUs			
smm-db	S0: 10 DTUs			
tasks-db	S0: 10 DTUs			
Web-db	S2: 50 DTUs			

EXM Dispatch settings:

- NumberThreads = 40
- MaxGenerationThreads = 40
- DispatchEnqueueBatchSize = 300
- DispatchEnqueueThreadsNumber = 4
- EXM.DispatchBatchSize = 8

Test scenarios

In a single marketing campaign, you can use only one email template, and we need to test EXM performance for various email sizes. We must therefore repeat these scenarios for each email template.

In total, we have 5 templates for email messages of 50 KB, 75 KB, 100 KB, 200 KB, and 400 KB.

Phase I

- Create a contact list based on a file with contacts (100K Contacts).
- Create regular email campaigns for each size based on the corresponding template.

- Include matching contact list to this campaign.
- Run all common campaigns one by one.
- Create personalized email campaigns for each size based on the corresponding template.
- Include matching contact list to this campaign.
- Run all personalized campaigns one by one.

Phase II

- Open the email message.
- Click on email links according to the load profile.

Test infrastructure configuration

Monitoring configuration

Grafana:

- Azure Monitor Data Sources Plugin for collecting Azure Web Apps metrics
- InfluxDB for collecting metrics from JMeter

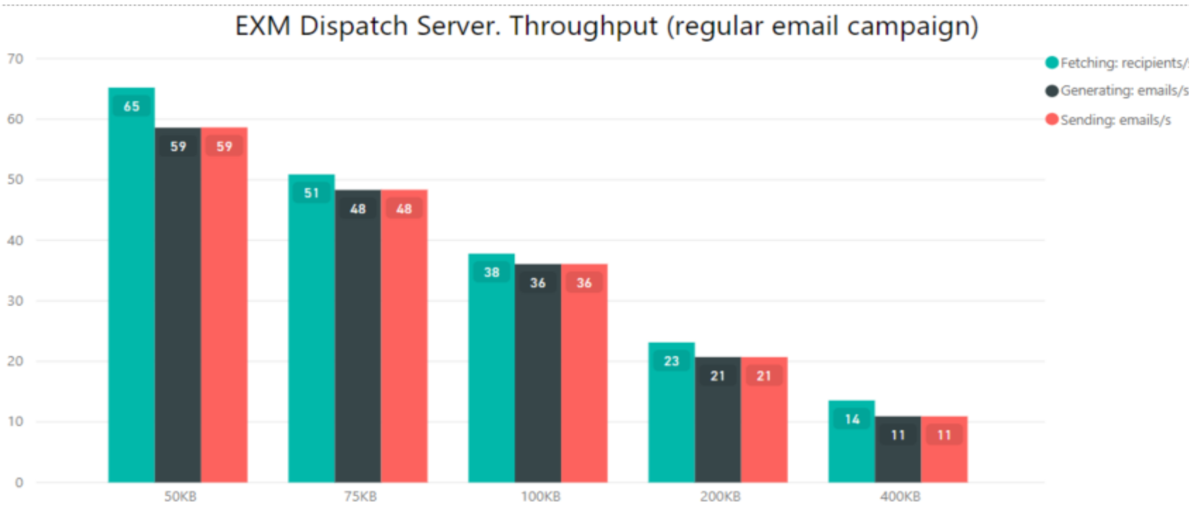
Performance metrics

Metric	Description
SPEED	Number of email messages per second during dispatch
	Number of email messages per hour during dispatch
TIME SPENT	Opens processed
	Clicks processed
	Unsubscribes processed
LOAD, RESOURCE USAGE	CPU cores usage
	RAM usage
	SQL load per DB
	SQL DBs sizes
	Azure Search metrics
OTHER	Warnings, errors logged during dispatch

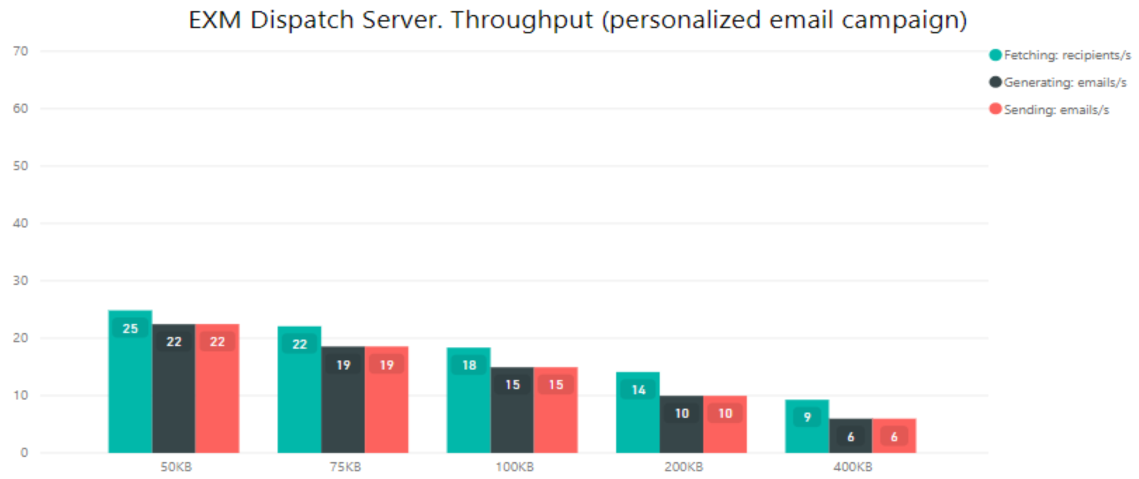
Phase I. Sending email messages

Test summary

Let us suppose that we spend about 8 hours a day sending out marketing campaigns. Usually, for this action, the time interval with the lowest user activity should be chosen. Thus, we can process:



Regular email campaign				
Email size, KB	Sending emails/s	Sending emails/hour	Sending emails/day	Number of days to process 5M emails
50	59	212 400	1 699 200	3
75	48	172 800	1 382 400	4
100	36	129 600	1 036 800	5
200	21	75 600	604 800	9
400	11	39 600	316 800	16



Personalized email campaign				
Email size, KB	Sending emails/s	Sending emails/hour	Sending emails/day	Number of days to process 5M emails
50	22	79 200	633 600	8
75	19	68 400	547 200	10
100	15	54 000	432 000	12
200	10	36 000	288 000	18
400	6	21 600	172 800	29

Larger emails increase CPU consumption:

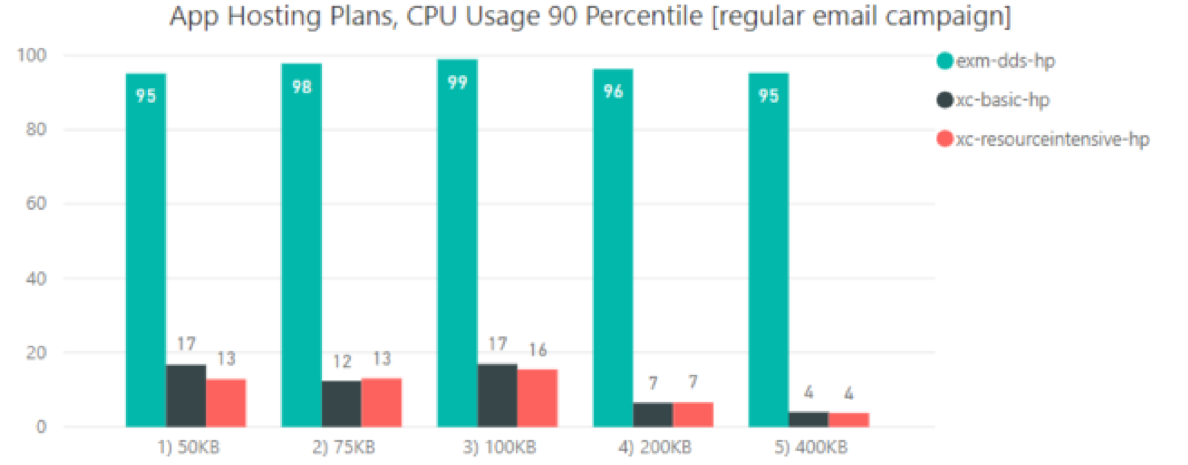
Regular Email Campaign

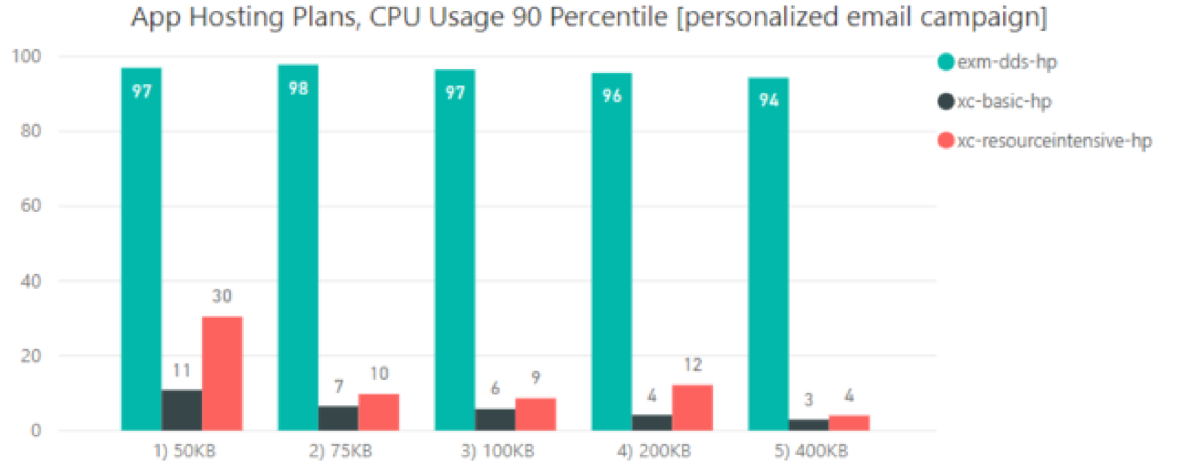
E-Mail size/CPU %	50KB	75 KB	100 KB	200 KB	400 KB
App HP EXM DDS	95.17	97.80	99.00	96.32	95.33
SQL DB EXM Master	25.60	22.00	18.00	14.00	12.00
SQL DB MA	34.20	23.00	16.00	10.00	5.00
SQL DB Pools	18.00	16.00	13.00	10.00	7.00
SQL DB Shard0	61.20	55.10	52.60	62.00	62.00
SQL DB Shard1	65.60	60.00	58.00	56.00	45.00

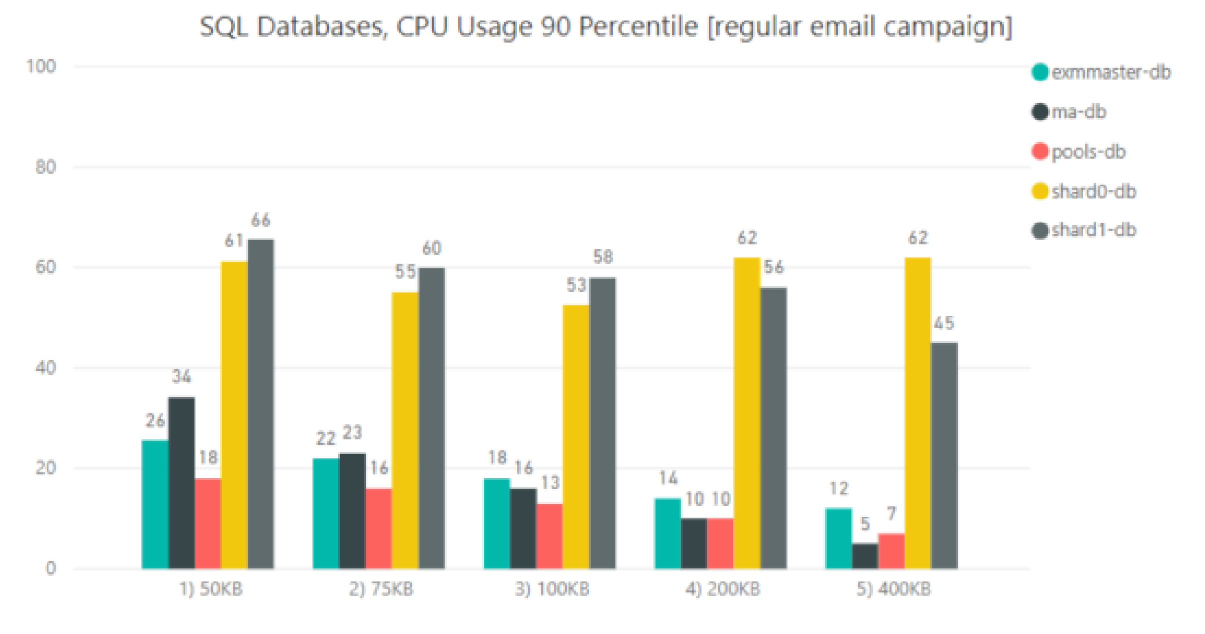
Personalized Email Campaign

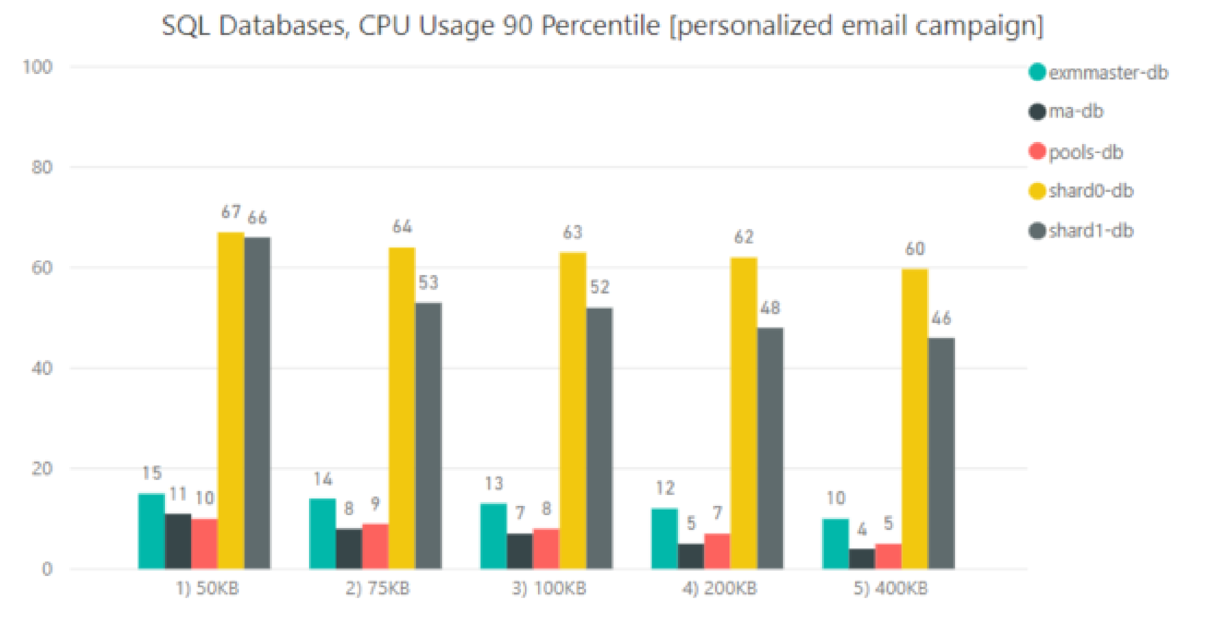
E-Mail size/CPU %	50KB	75 KB	100 KB	200 KB	400 KB
App HP EXM DDS	97.00	97.80	96.50	95.67	94.37
SQL DB EXM Master	15.00	14.00	13.00	12.00	10.00
SQL DB MA	11.00	8.00	7.00	5.00	4.00
SQL DB Pools	10.00	9.00	8.00	7.00	5.00
SQL DB Shard0	67.00	64.00	63.00	62.00	59.70
SQL DB Shard1	66.00	53.00	52.00	48.00	46.00

DDS server CPU consumption is kept at 95% regardless of email message size. CPU is a bottleneck of the EXM dispatching






SQL Databases, CPU Usage 90 Percentile [regular email campaign]
exmmaster-db
ma-db
pools-db
shard0-db
shard1-db
100
80
60
40
20
0
1) 50KB: 26, 34, 18, 61, 66
2) 75KB: 22, 23, 16, 55, 60
3) 100KB: 18, 16, 13, 53, 58
4) 200KB: 14, 10, 10, 62, 56
5) 400KB: 12, 5, 7, 62, 45

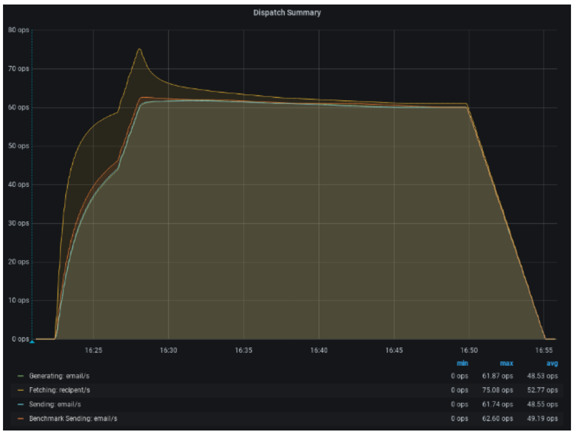

SQL Databases, CPU Usage 90 Percentile [personalized email campaign]
exmmaster-db
ma-db
pools-db
shard0-db
shard1-db
100
80
60
40
20
0
1) 50KB: 15, 11, 10, 67, 66
2) 75KB: 14, 8, 9, 64, 53
3) 100KB: 13, 7, 8, 63, 52
4) 200KB: 12, 5, 7, 62, 48
5) 400KB: 10, 4, 5, 60, 46

Regular email campaign

EXM dispatch metrics

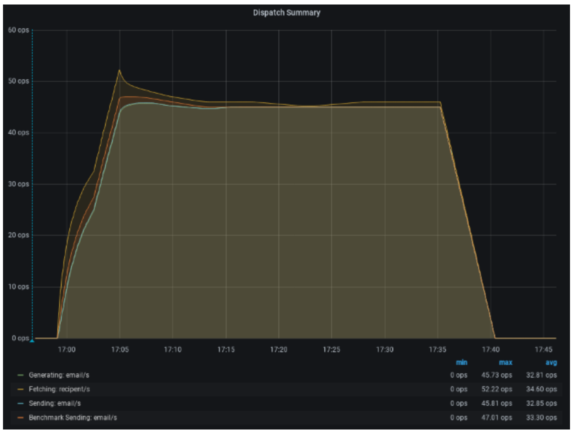
Dispatch summary

50KB Regular email campaign



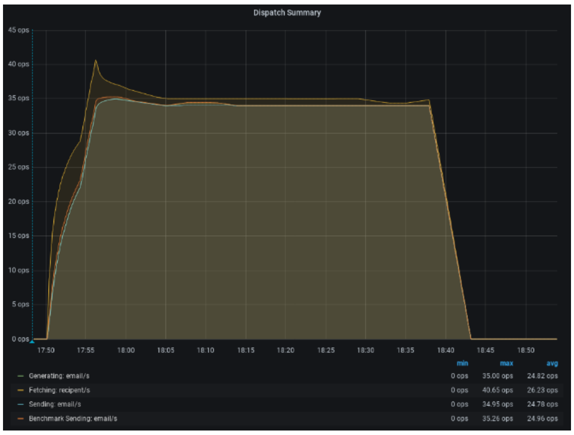
	min	max	avg
Generating: email/s	0 ops	61.87 ops	48.53 ops
Fetching: recipent/s	0 ops	75.08 ops	52.77 ops
Sending: email/s	0 ops	61.74 ops	48.55 ops
Benchmark Sending: email/s	0 ops	62.60 ops	49.19 ops

75KB Regular email campaign



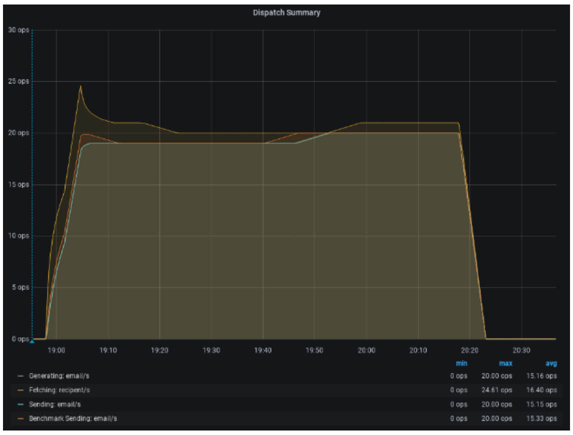
	min	max	avg
Generating: email/s	0 ops	45.73 ops	32.81 ops
Fetching: recipent/s	0 ops	52.22 ops	34.60 ops
Sending: email/s	0 ops	45.81 ops	32.85 ops
Benchmark Sending: email/s	0 ops	47.01 ops	33.30 ops

100KB Regular email campaign



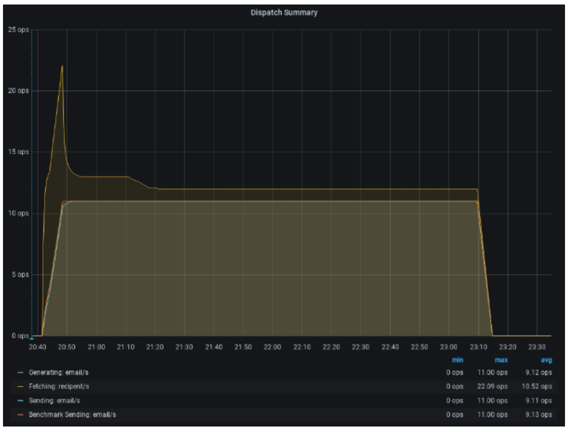
	min	max	avg
Generating: email/s	0 ops	35.00 ops	24.82 ops
Fetching: recipent/s	0 ops	40.65 ops	26.23 ops
Sending: email/s	0 ops	34.95 ops	24.78 ops
Benchmark Sending: email/s	0 ops	35.26 ops	24.96 ops

200KB Regular email campaign



	min	max	avg
Generating: email/s	0 ops	20.00 ops	15.16 ops
Fetching: recipent/s	0 ops	24.61 ops	16.40 ops
Sending: email/s	0 ops	20.00 ops	15.15 ops
Benchmark Sending: email/s	0 ops	20.00 ops	15.33 ops

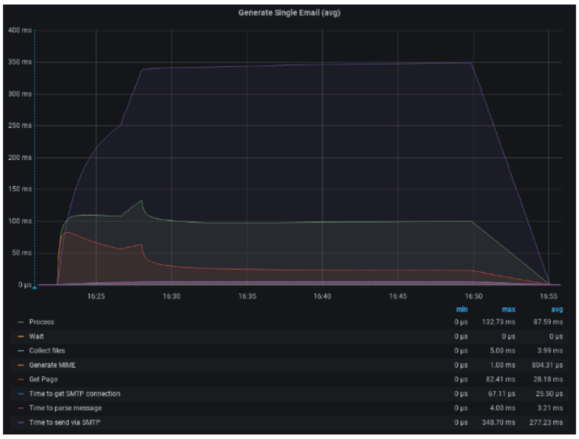
400KB Regular email campaign



	min	max	avg
Generating: email/s	0 ops	11.00 ops	9.12 ops
Fetching: recipent/s	0 ops	22.09 ops	10.52 ops
Sending: email/s	0 ops	11.00 ops	9.11 ops
Benchmark Sending: email/s	0 ops	11.00 ops	9.13 ops

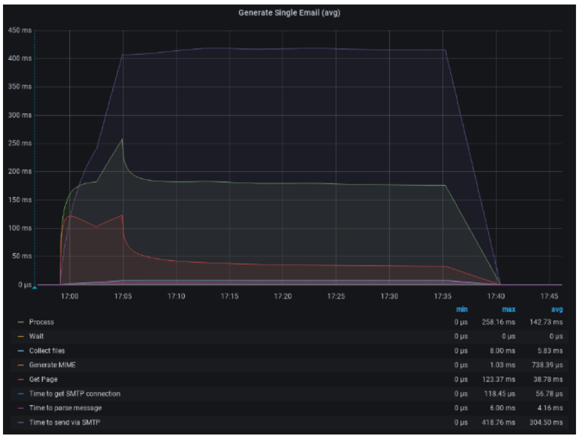
Generate single email campaign

50KB regular email campaign



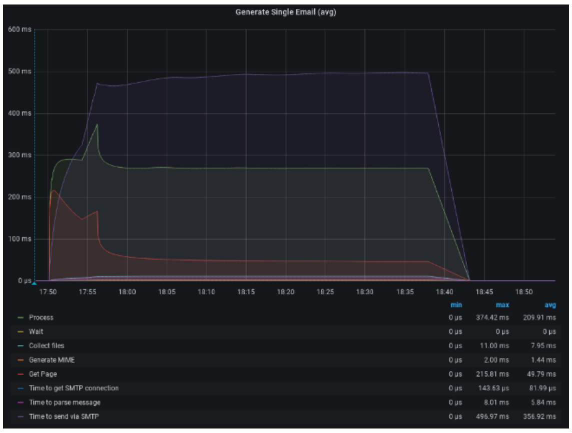
	min	max	avg
Process	0 µs	132.73 ms	87.59 ms
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	5.00 ms	3.99 ms
Generate MIME	0 µs	1.00 ms	804.31 µs
Get Page	0 µs	82.41 ms	28.18 ms
Time to get SMTP connection	0 µs	67.11 µs	25.50 µs
Time to parse message	0 µs	4.00 ms	3.21 ms
Time to send via SMTP	0 µs	348.70 ms	277.23 ms

75KB regular email campaign



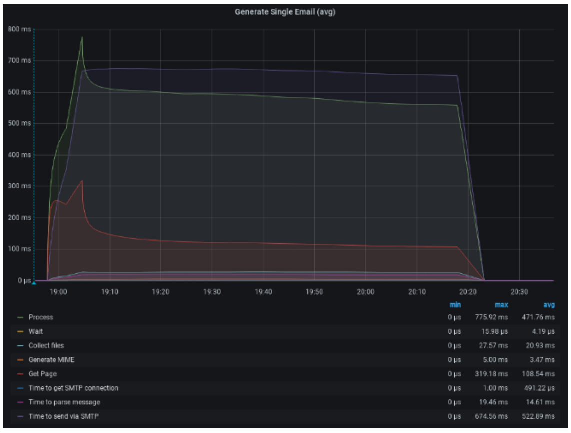
	min	max	avg
Process	0 µs	258.16 ms	142.73 ms
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	8.00 ms	5.83 ms
Generate MIME	0 µs	1.03 ms	738.39 µs
Get Page	0 µs	123.37 ms	38.78 ms
Time to get SMTP connection	0 µs	118.45 µs	56.78 µs
Time to parse message	0 µs	6.00 ms	4.16 ms
Time to send via SMTP	0 µs	418.76 ms	304.50 ms

100KB regular email campaign



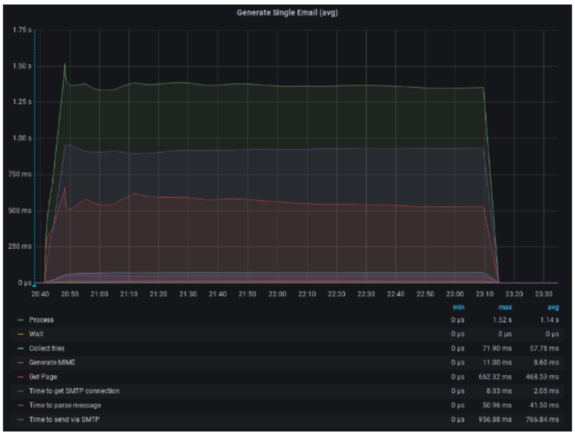
	min	max	avg
Process	0 µs	374.42 ms	209.91 ms
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	11.00 ms	7.95 ms
Generate MIME	0 µs	2.00 ms	1.44 ms
Get Page	0 µs	215.81 ms	49.79 ms
Time to get SMTP connection	0 µs	143.63 µs	81.99 µs
Time to parse message	0 µs	8.01 ms	5.84 ms
Time to send via SMTP	0 µs	496.97 ms	356.92 ms

200KB regular email campaign



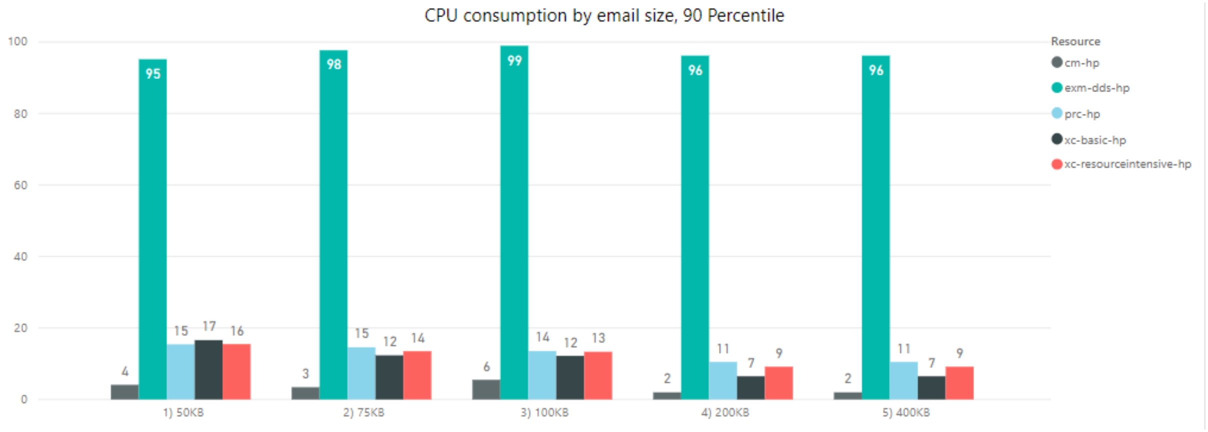
	min	max	avg
Process	0 µs	775.92 ms	471.76 ms
Wait	0 µs	15.98 µs	4.19 µs
Collect files	0 µs	27.57 ms	20.93 ms
Generate MIME	0 µs	5.00 ms	3.47 ms
Get Page	0 µs	319.18 ms	108.54 ms
Time to get SMTP connection	0 µs	1.00 ms	491.22 µs
Time to parse message	0 µs	19.46 ms	14.61 ms
Time to send via SMTP	0 µs	674.56 ms	522.89 ms

400KB regular email campaign

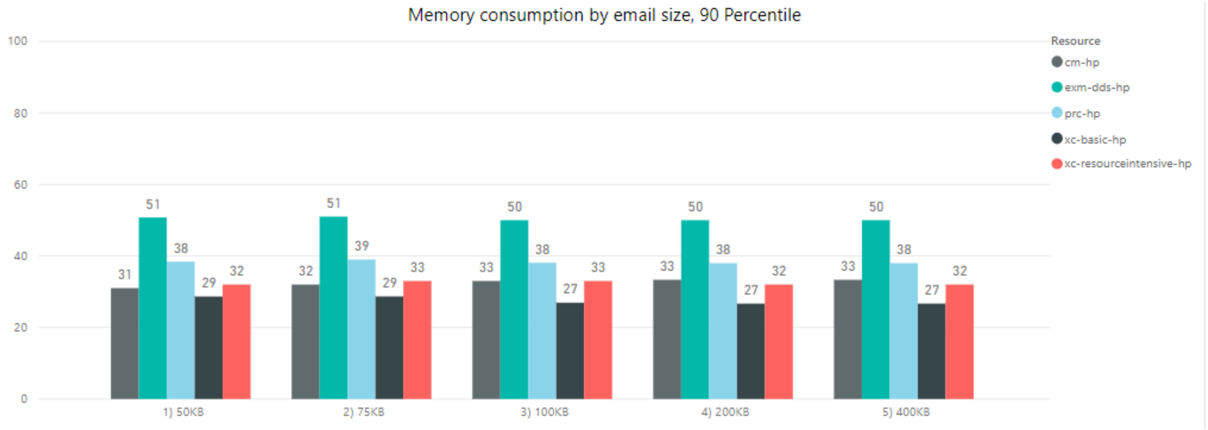


	min	max	avg
Process	0 µs	1.52 s	1.14 s
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	71.90 ms	57.78 ms
Generate MIME	0 µs	11.00 ms	8.60 ms
Get Page	0 µs	662.32 ms	468.53 ms
Time to get SMTP connection	0 µs	8.03 ms	2.05 ms
Time to parse message	0 µs	50.96 ms	41.50 ms
Time to send via SMTP	0 µs	956.88 ms	766.84 ms

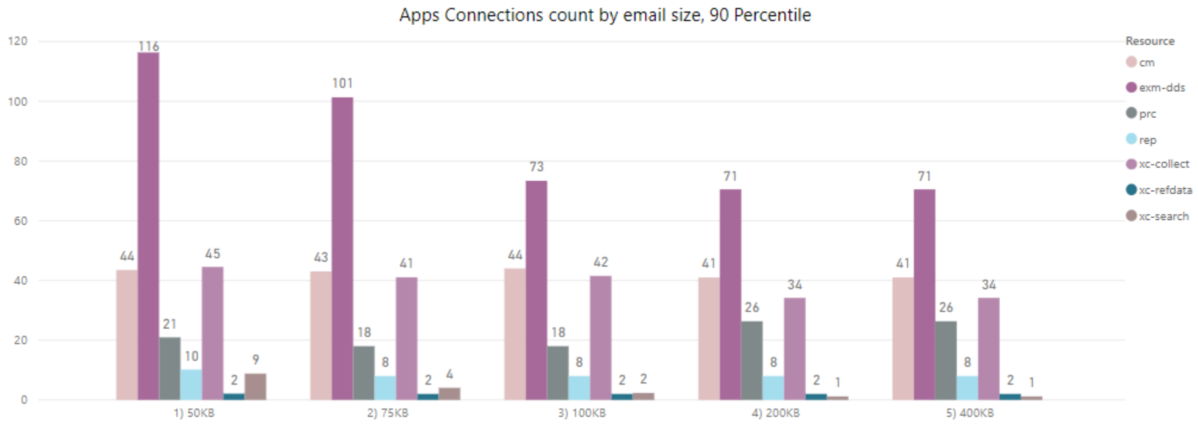
CPU – app service hosting plans



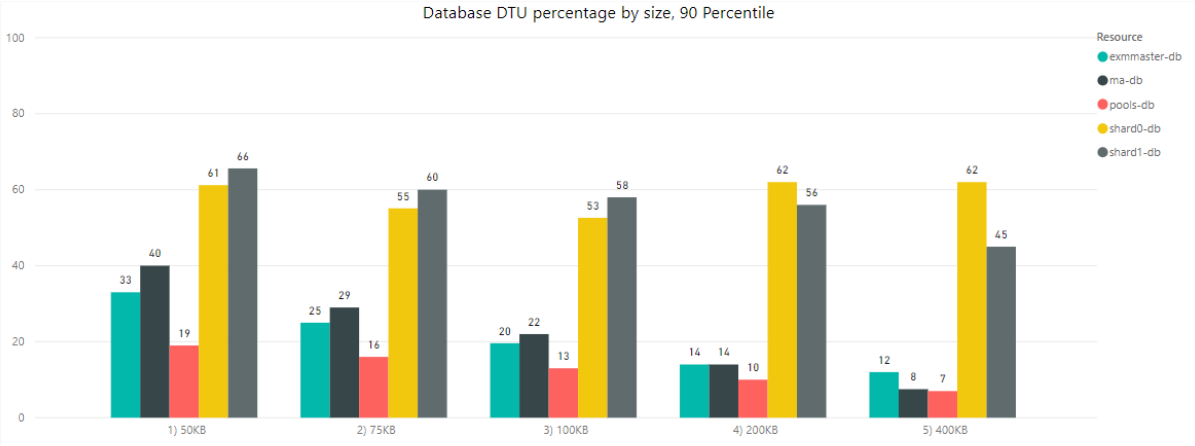
Memory - app service hosting plans



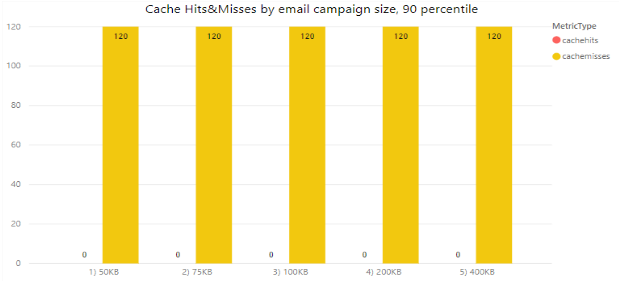
Connections - app services

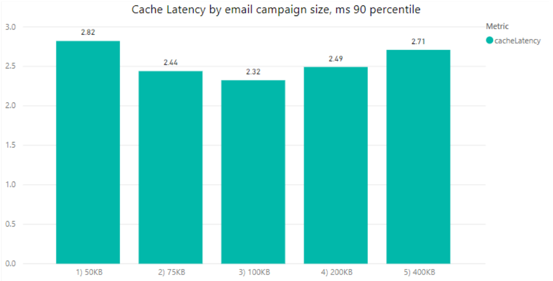


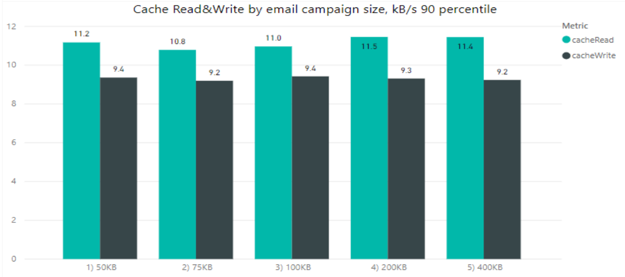
Azure SQL databases – DTU percentage

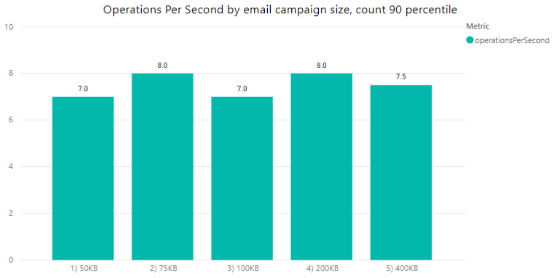


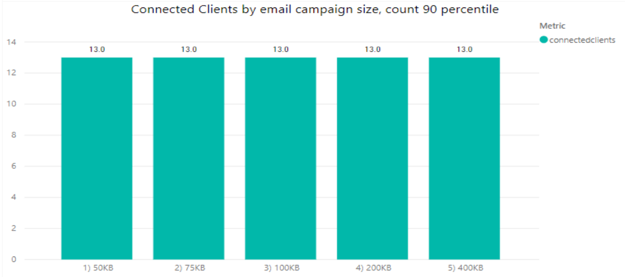
Azure Cache - Redis Metrics

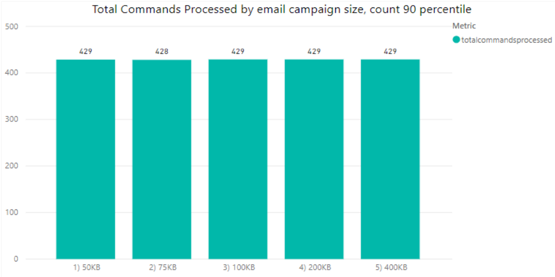


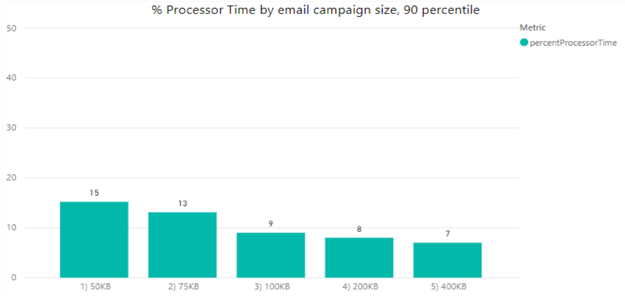


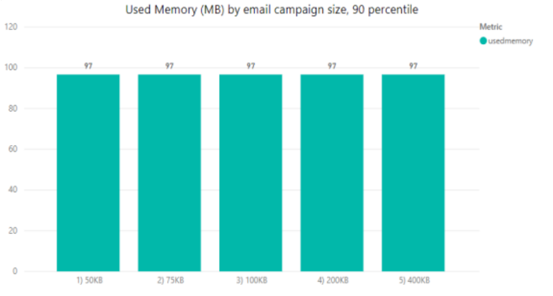










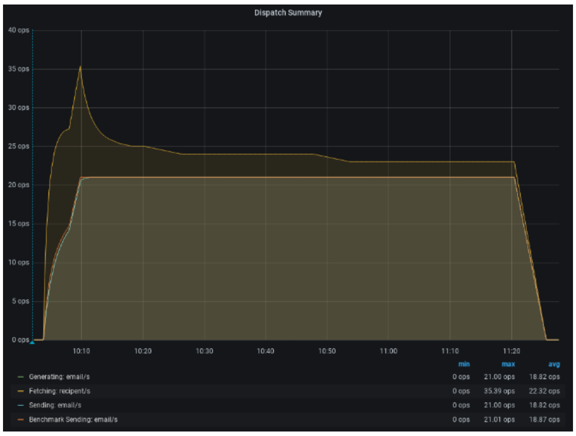


Personalized email campaign

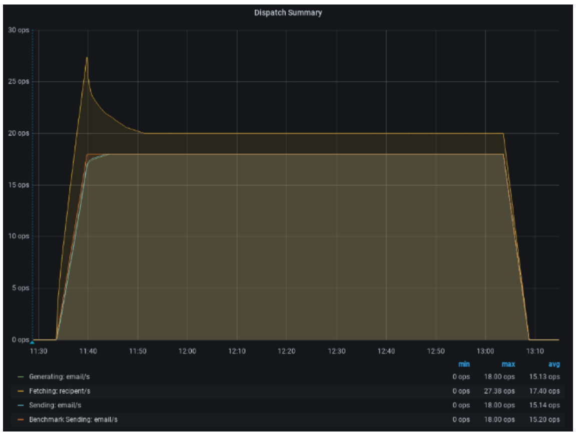
EXM dispatch metrics

Dispatch summary

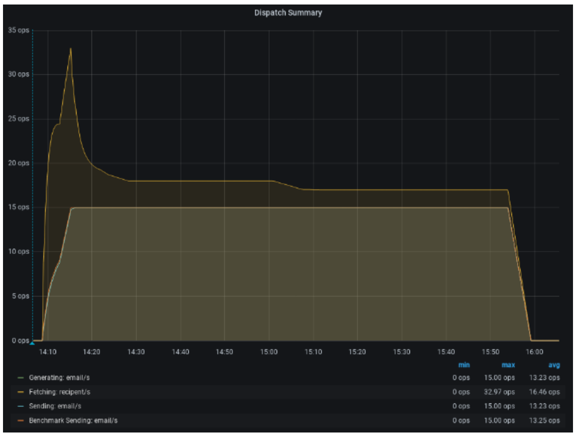
50KB Personalized email campaign



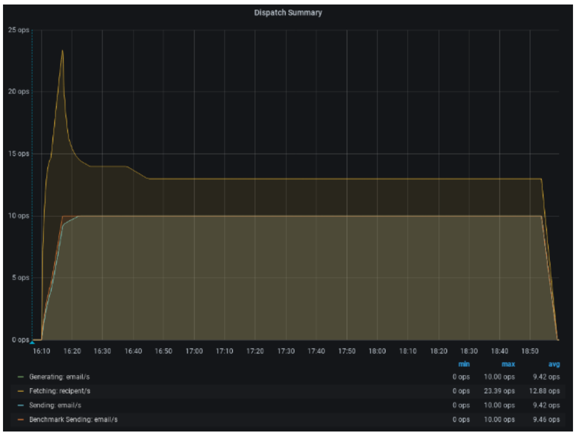
75KB Personalized email campaign



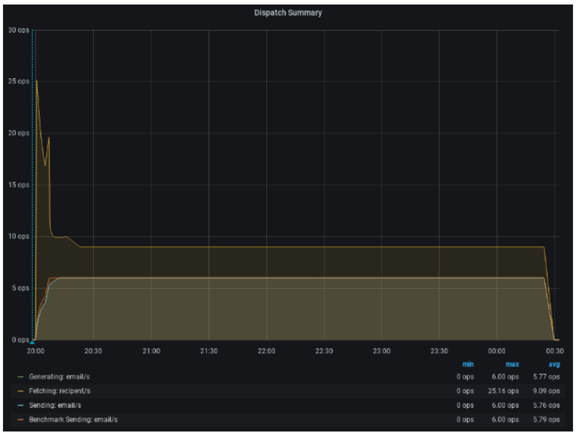
100KB Personalized email campaign



200KB Personalized email campaign

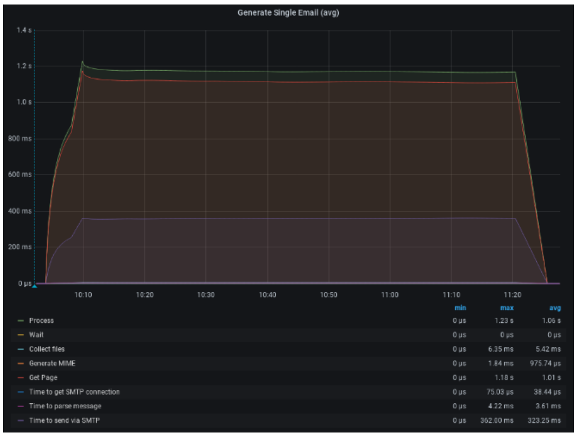


400KB Personalized email campaign



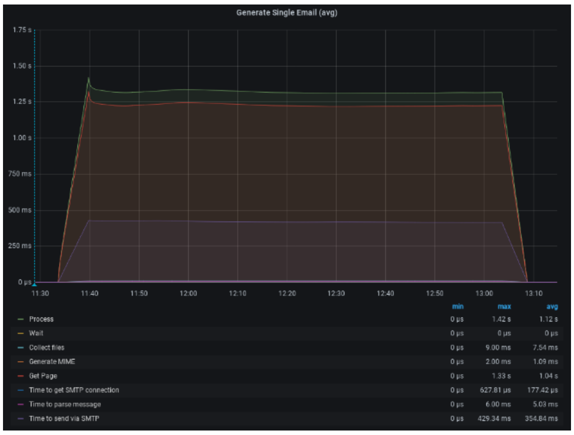
Generate single email message

50KB Personalized email campaign



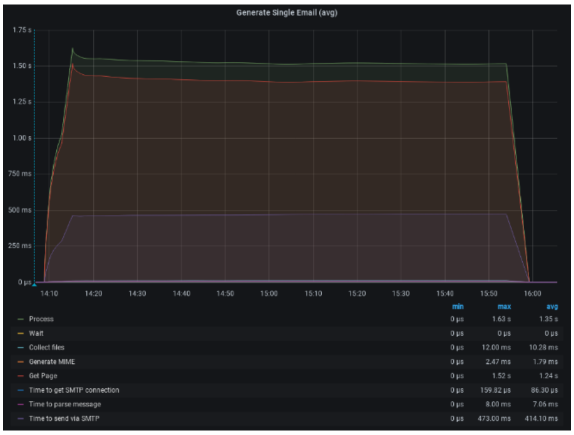
	min	max	avg
Process	0 µs	1.23 s	1.06 s
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	6.35 ms	5.42 ms
Generate MIME	0 µs	1.84 ms	975.74 µs
Get Page	0 µs	1.18 s	1.01 s
Time to get SMTP connection	0 µs	75.03 µs	38.44 µs
Time to parse message	0 µs	4.22 ms	3.61 ms
Time to send via SMTP	0 µs	362.00 ms	323.25 ms

75KB Personalized email campaign



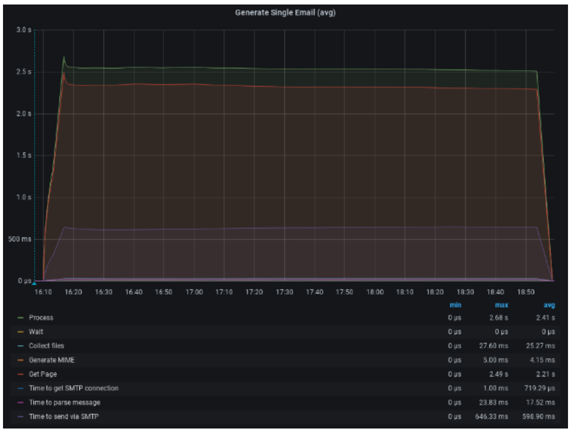
	min	max	avg
Process	0 µs	1.42 s	1.12 s
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	9.00 ms	7.54 ms
Generate MIME	0 µs	2.00 ms	1.09 ms
Get Page	0 µs	1.33 s	1.04 s
Time to get SMTP connection	0 µs	627.81 µs	177.42 µs
Time to parse message	0 µs	6.00 ms	5.03 ms
Time to send via SMTP	0 µs	429.34 ms	354.84 ms

100KB Personalized email campaign



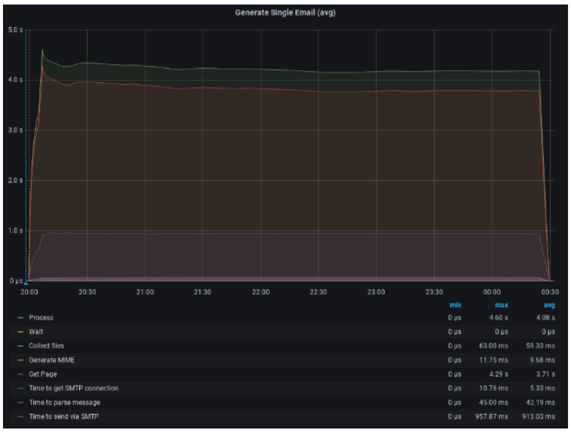
	min	max	avg
Process	0 µs	1.63 s	1.35 s
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	12.00 ms	10.28 ms
Generate MIME	0 µs	2.47 ms	1.79 ms
Get Page	0 µs	1.52 s	1.24 s
Time to get SMTP connection	0 µs	159.82 µs	86.30 µs
Time to parse message	0 µs	8.00 ms	7.06 ms
Time to send via SMTP	0 µs	473.00 ms	414.10 ms

200KB Personalized email campaign



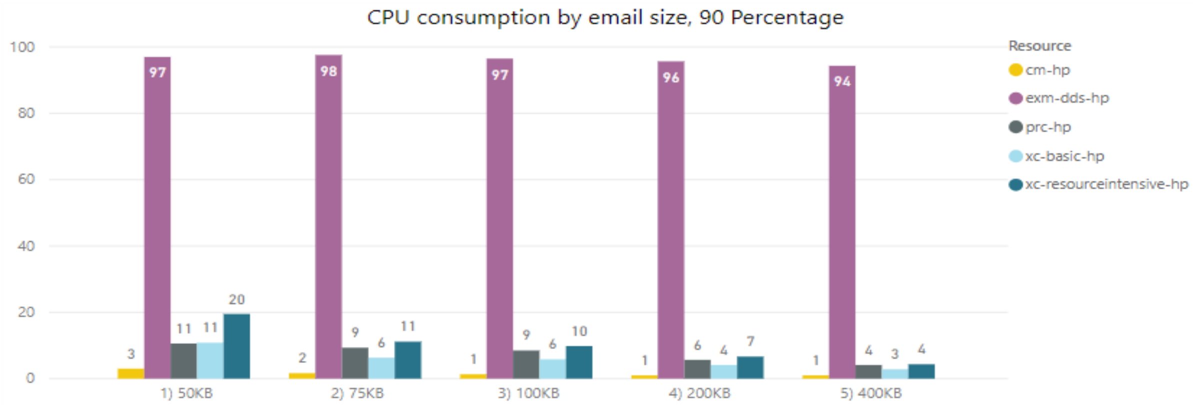
	min	max	avg
Process	0 µs	2.68 s	2.41 s
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	27.60 ms	25.27 ms
Generate MIME	0 µs	5.00 ms	4.15 ms
Get Page	0 µs	2.49 s	2.21 s
Time to get SMTP connection	0 µs	1.00 ms	719.29 µs
Time to parse message	0 µs	23.83 ms	17.52 ms
Time to send via SMTP	0 µs	646.33 ms	598.90 ms

400KB Personalized email campaign

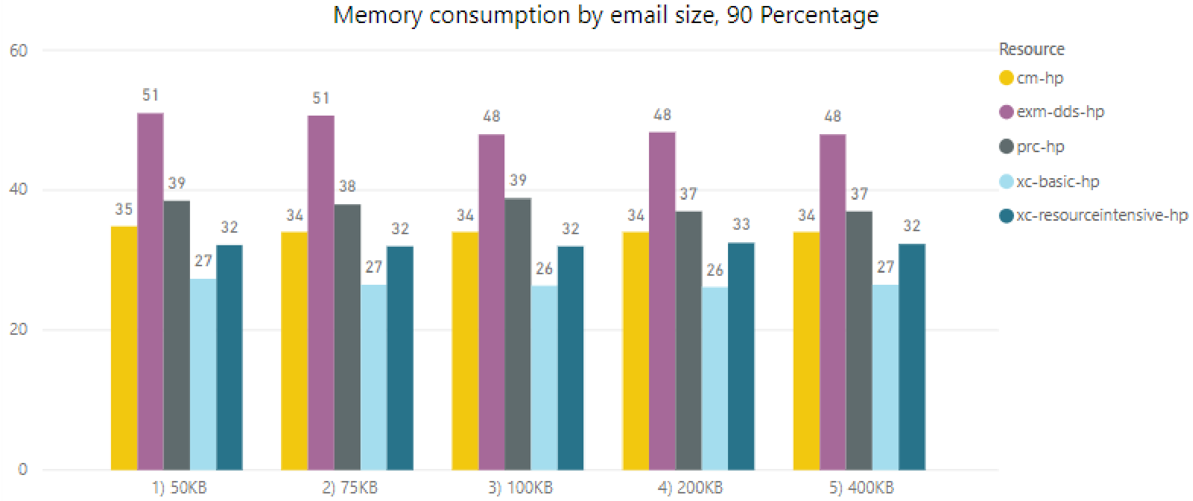


	min	max	avg
Process	0 µs	4.60 s	4.08 s
Wait	0 µs	0 µs	0 µs
Collect files	0 µs	63.00 ms	59.33 ms
Generate MIME	0 µs	11.75 ms	9.68 ms
Get Page	0 µs	4.29 s	3.71 s
Time to get SMTP connection	0 µs	10.76 ms	5.33 ms
Time to parse message	0 µs	45.00 ms	42.19 ms
Time to send via SMTP	0 µs	957.87 ms	913.03 ms

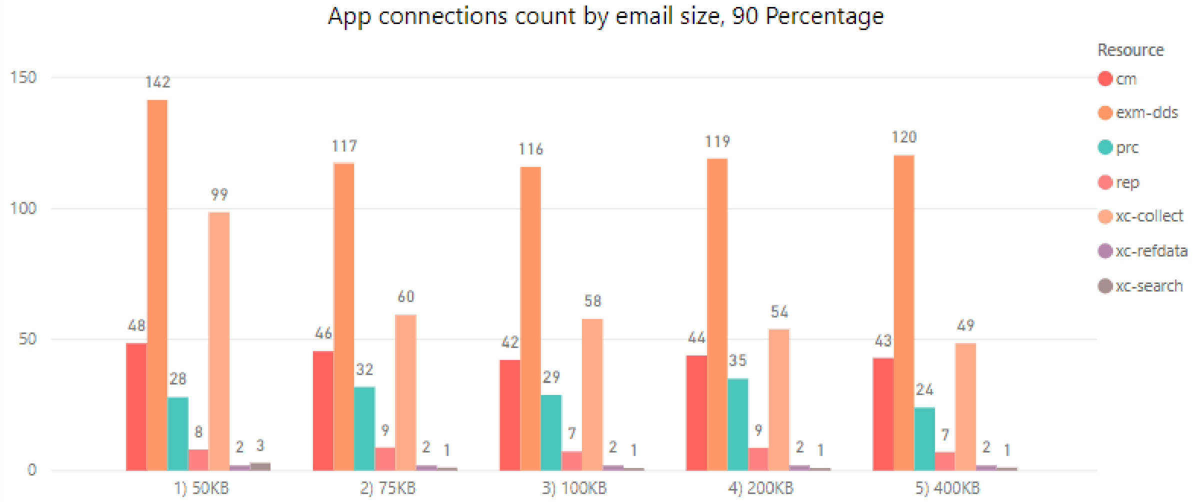
CPU – app service hosting plans



Memory - app service hosting plans

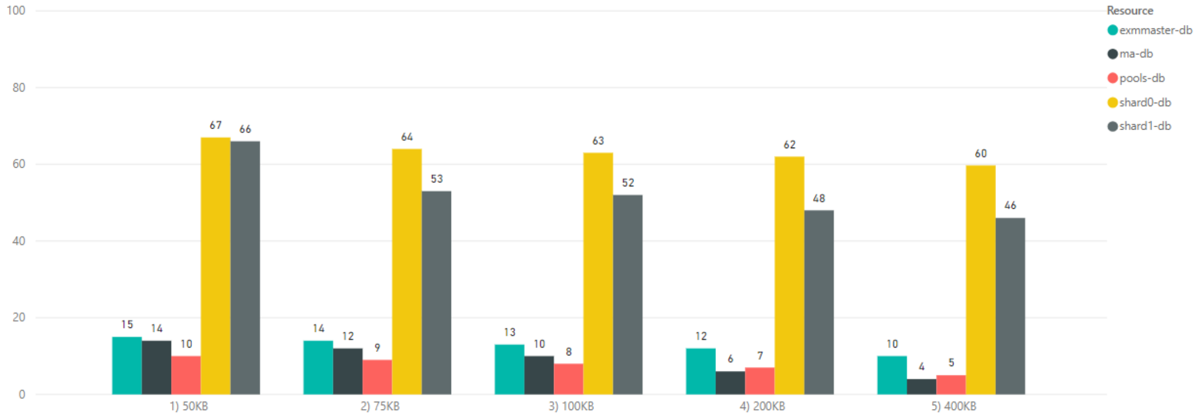


Connections - app services



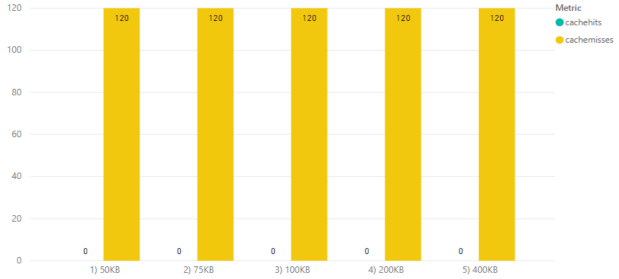
Azure SQL databases – DTU percentage

Database DTU percentage by size, 90 Percentile

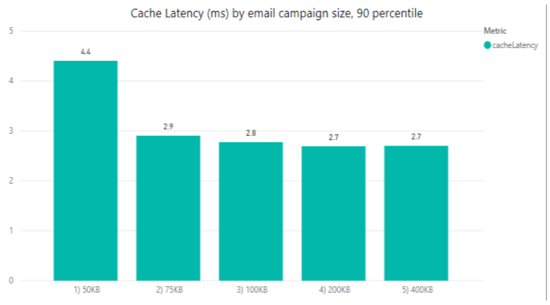


Azure Cache - Redis Metrics

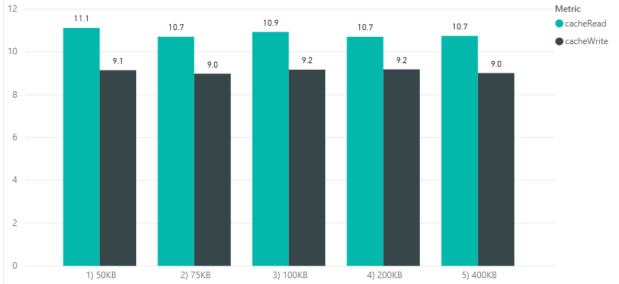
Cache Hits&Misses count by email campaign size, 90 percentile



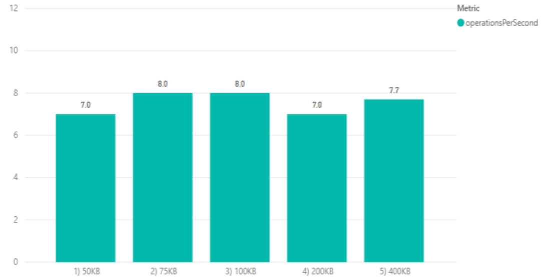
Cache Latency (ms) by email campaign size, 90 percentile



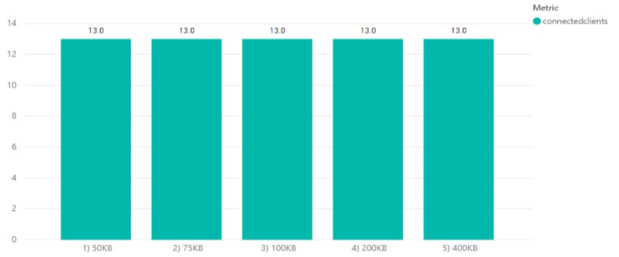
Cache Read&Write (kB/s) by email campaign size, 90 percentile



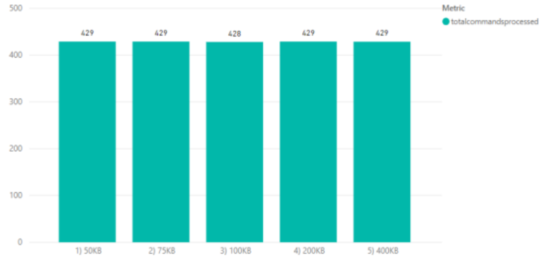
Operations per second (count) by email campaign size, 90 percentile



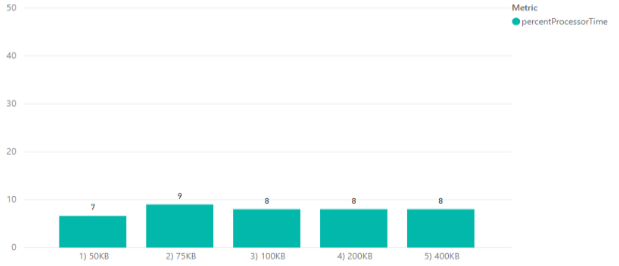
Connected clients (count) by email campaign size, 90 percentile



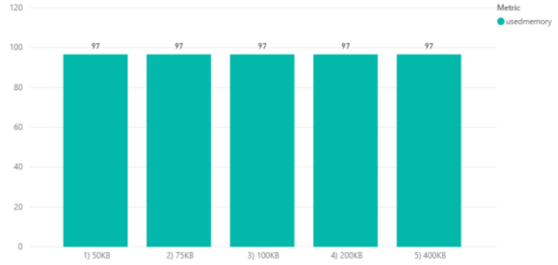
Total commands processed (count) by email campaign size, 90 percentile



Processor Time (%) by email campaign size, 90 percentile



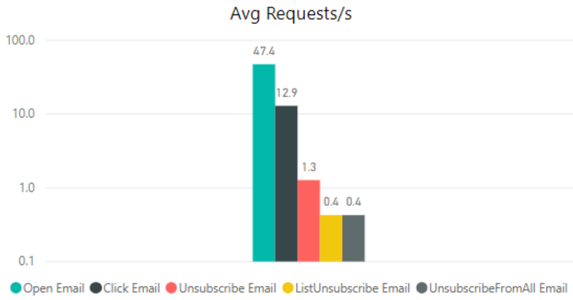
Used Memory (MB) by email campaign size, 90 percentile



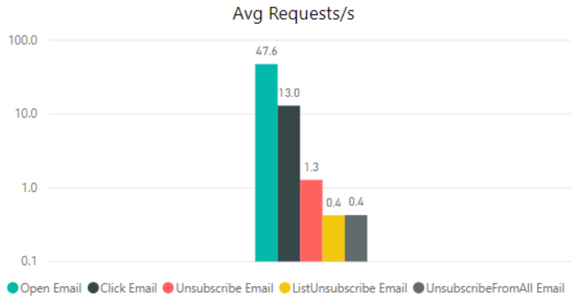
Phase II. Tracking email throughput

Test summary

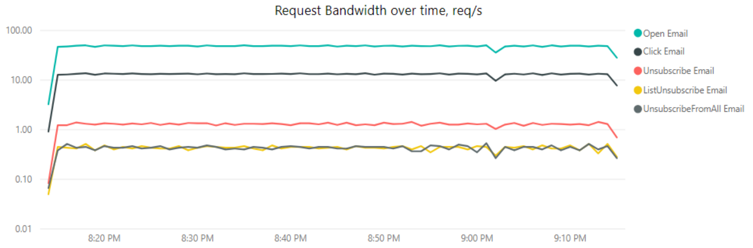
Regular email click throughput



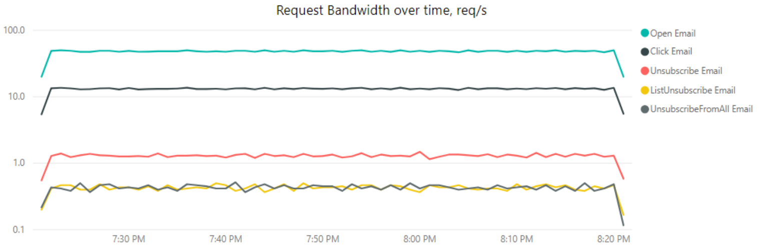
Personalized email click throughput



Regular emails



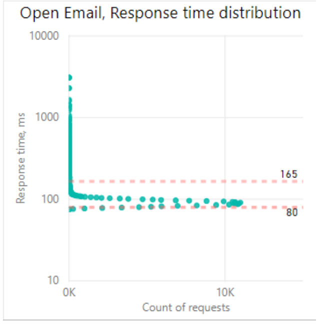
Personalized emails



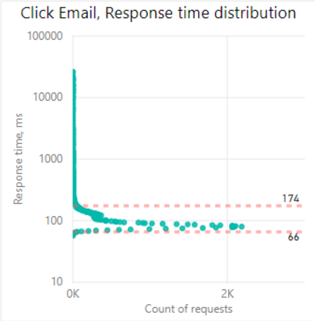
90 percentile throughput rates during the test was about 95 requests per second, which is about 342,000 requests per hour.

The response time for tested EXM email events is distributed from 441ms to 66ms. We can also conclude that the type of email company and the size of the message do not affect response time. More details can be seen as follows.

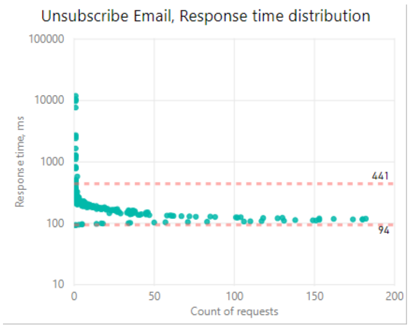
Distribution of response time for the following requests. Regular emails.



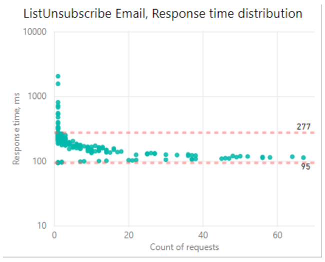
between 165–80ms



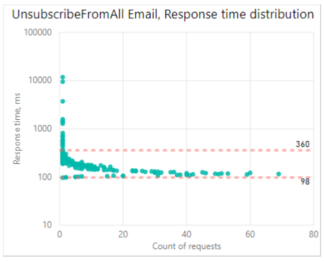
between 174 – 66ms



between 441 – 94ms

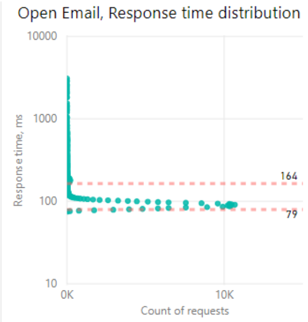


between 277–95ms

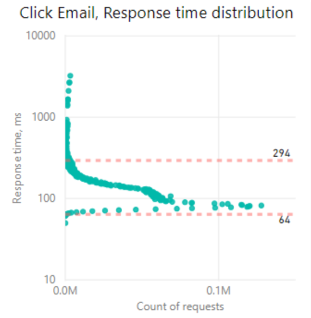


between 360 –98ms

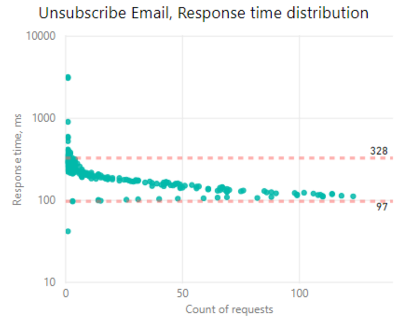
Distribution of response time for the following requests. Personalized emails.



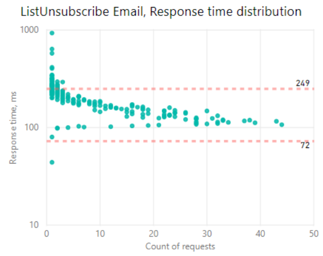
between 164–79ms



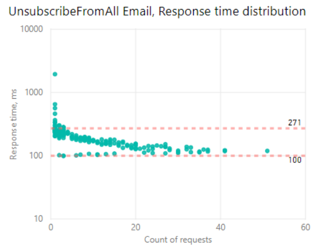
between 296 – 64ms



between 328 – 97ms



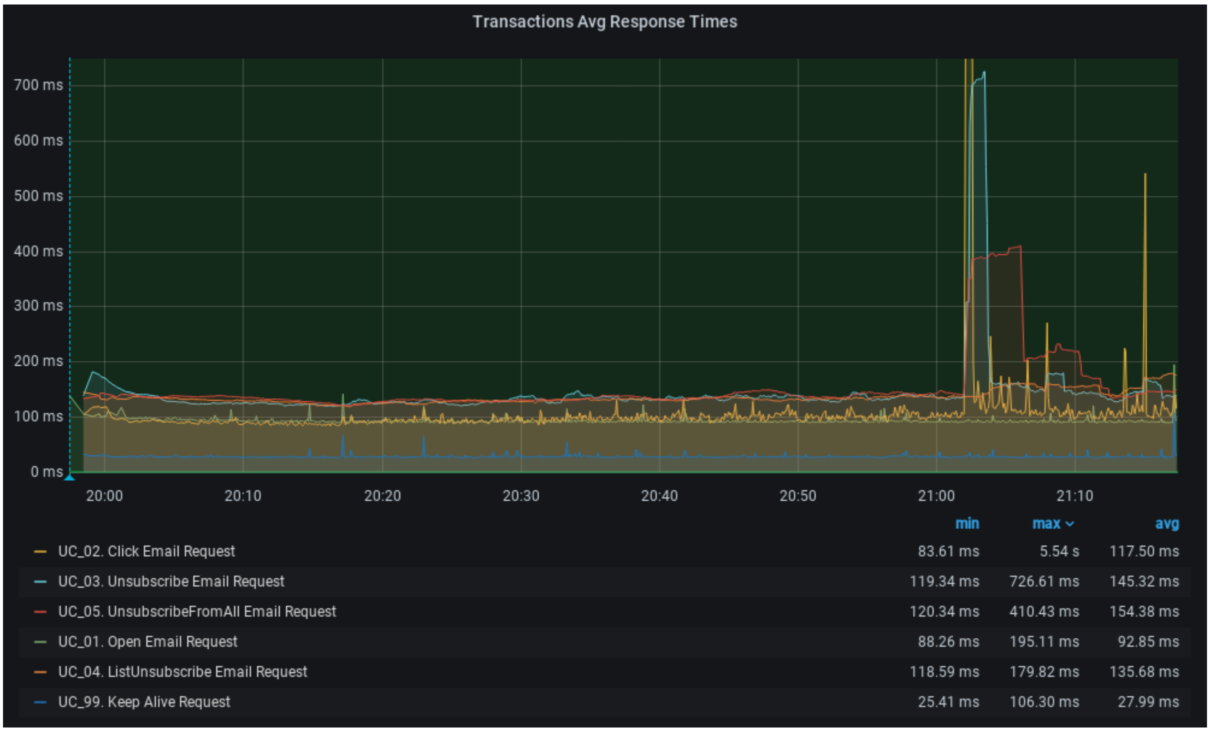
between 249–72ms



between 271 –100ms

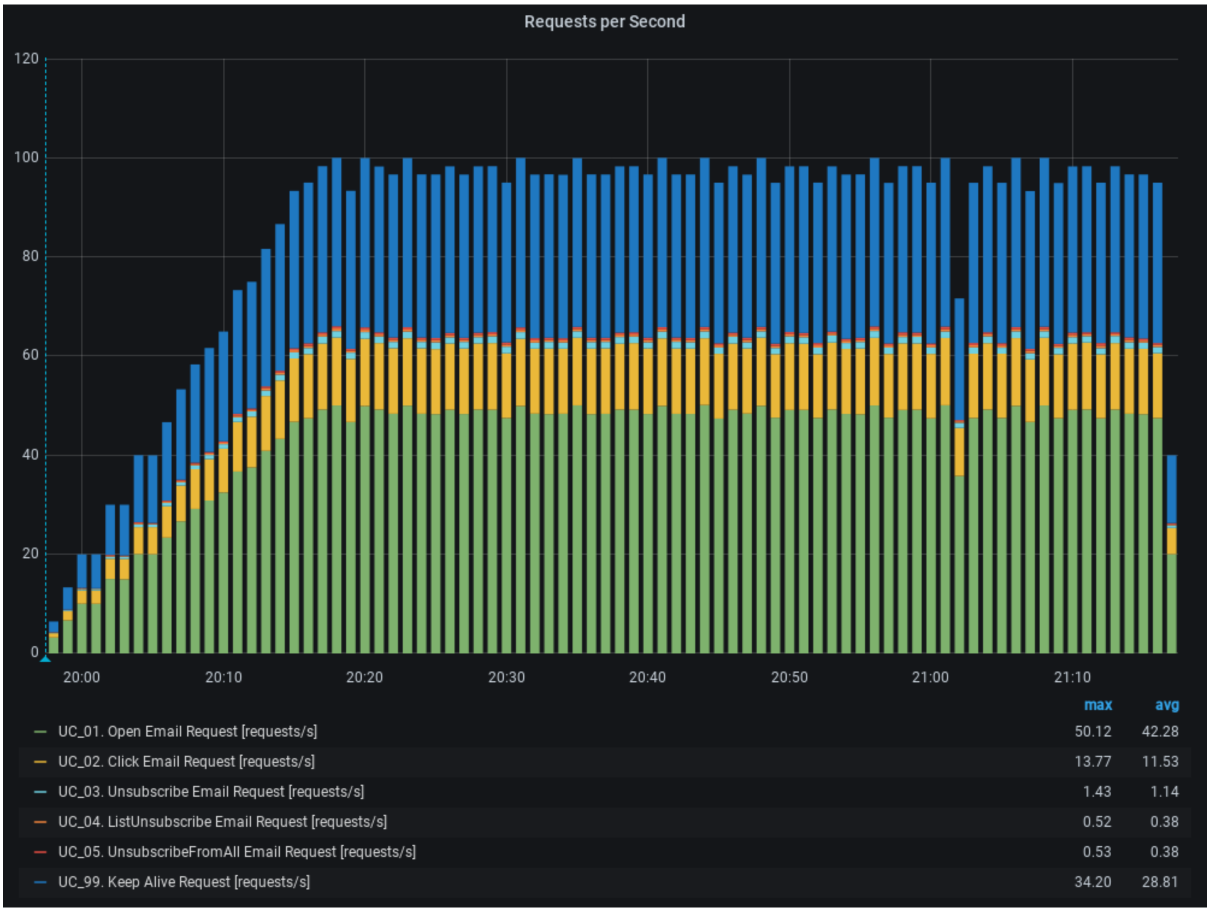
Regular email campaign

Response times



	min	max ⌄	avg
— UC_02. Click Email Request	83.61 ms	5.54 s	117.50 ms
— UC_03. Unsubscribe Email Request	119.34 ms	726.61 ms	145.32 ms
— UC_05. UnsubscribeFromAll Email Request	120.34 ms	410.43 ms	154.38 ms
— UC_01. Open Email Request	88.26 ms	195.11 ms	92.85 ms
— UC_04. ListUnsubscribe Email Request	118.59 ms	179.82 ms	135.68 ms
— UC_99. Keep Alive Request	25.41 ms	106.30 ms	27.99 ms

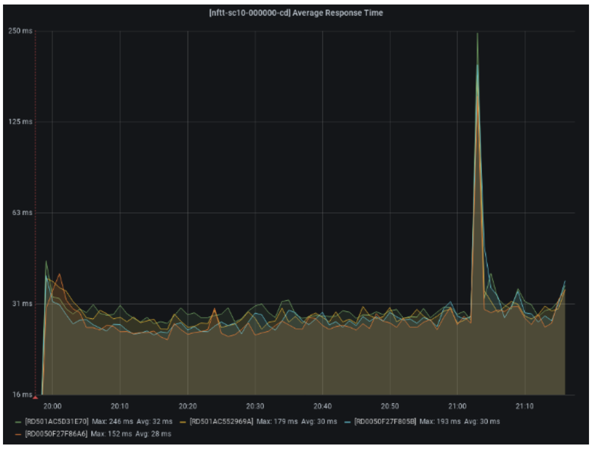
Requests per second



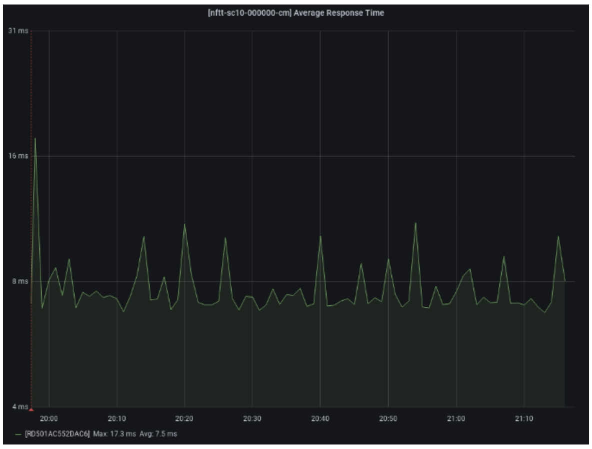
	max	avg
— UC_01. Open Email Request [requests/s]	50.12	42.28
— UC_02. Click Email Request [requests/s]	13.77	11.53
— UC_03. Unsubscribe Email Request [requests/s]	1.43	1.14
— UC_04. ListUnsubscribe Email Request [requests/s]	0.52	0.38
— UC_05. UnsubscribeFromAll Email Request [requests/s]	0.53	0.38
— UC_99. Keep Alive Request [requests/s]	34.20	28.81

Response times – app service plan

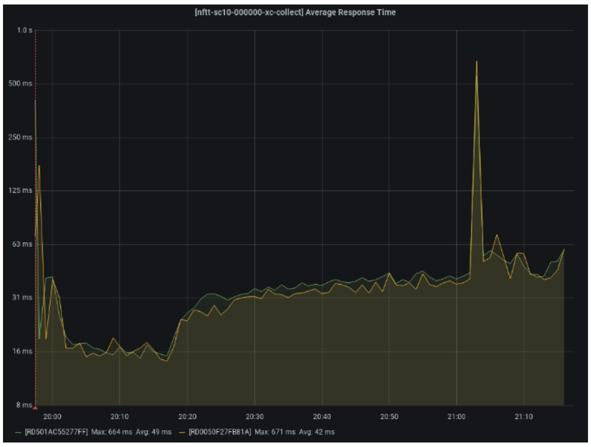
CD



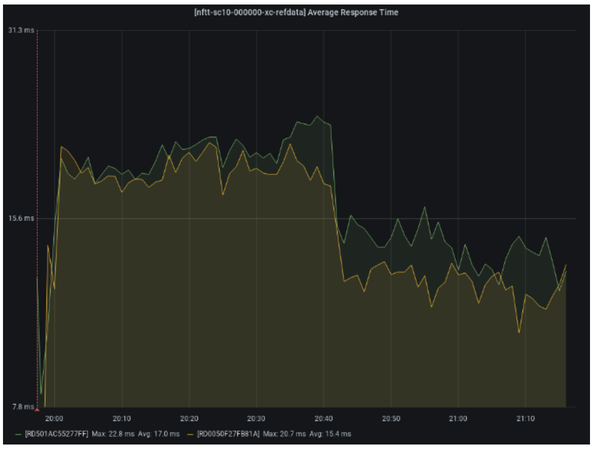
CM



XC-collection

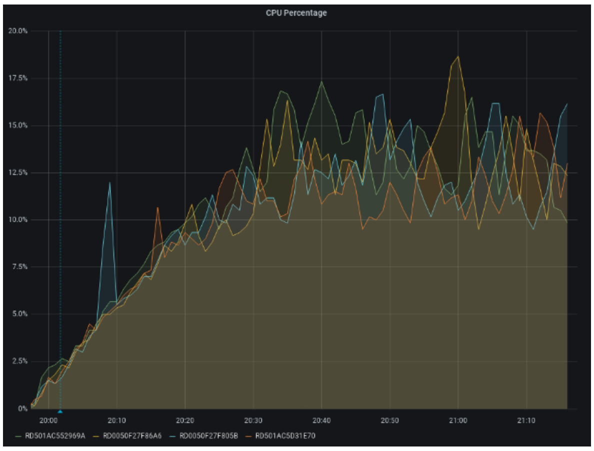


RefData

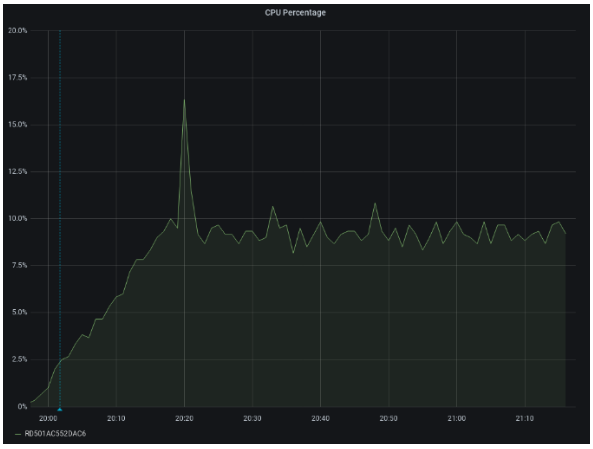


CPU – app service plan

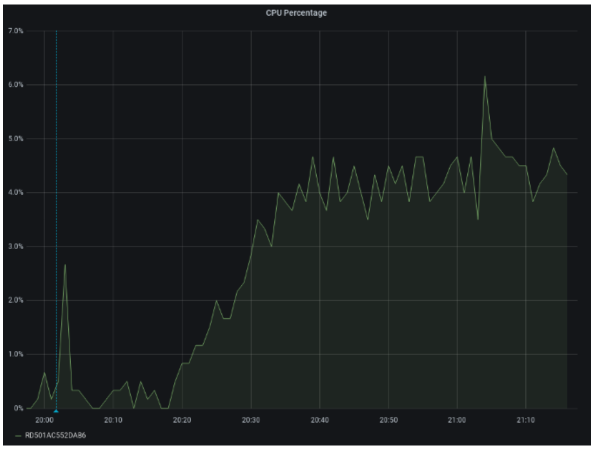
CD



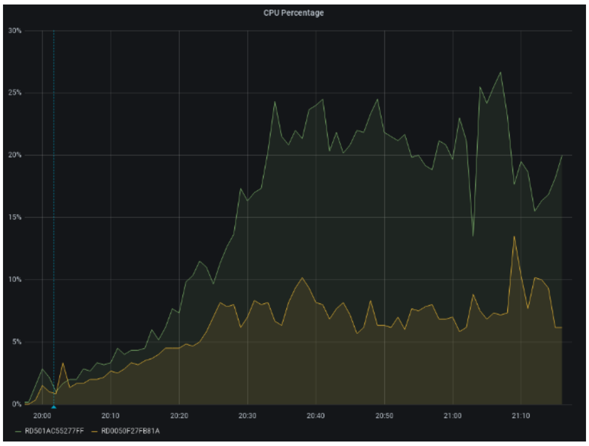
CM



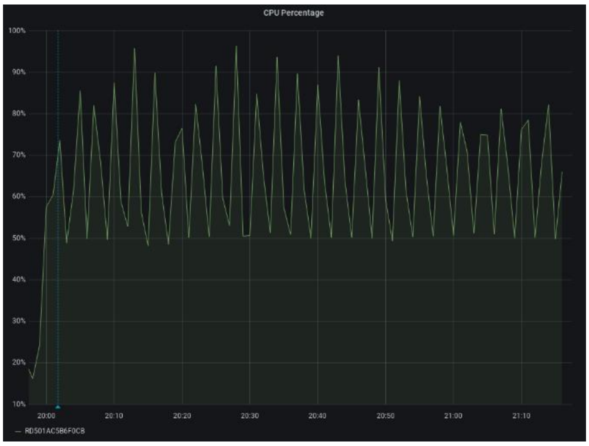
XC-Basic



XC Resource Intensive

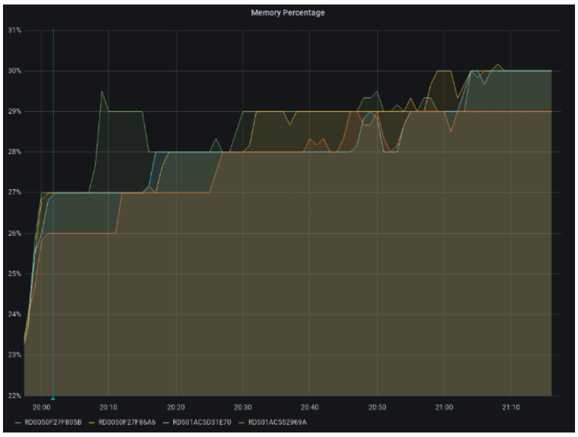


Prc

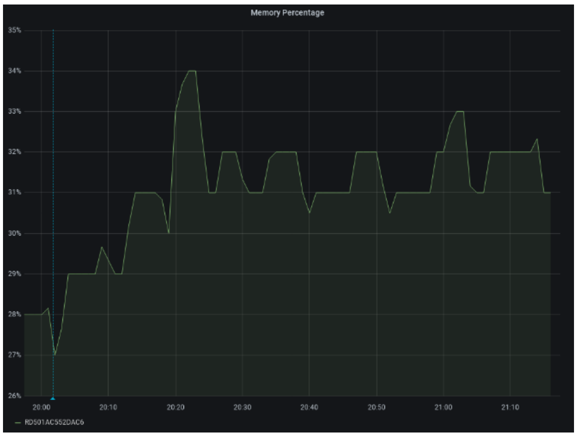


Memory – app service plan

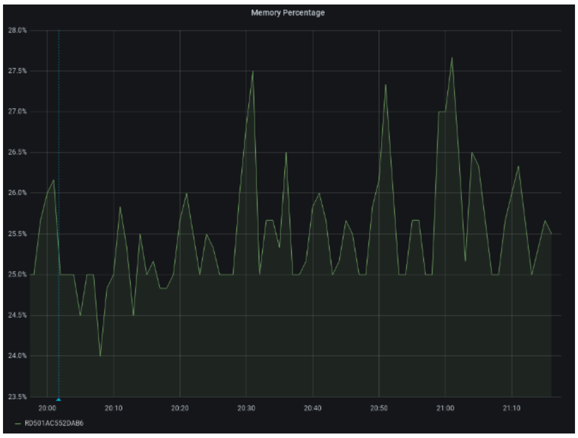
CD



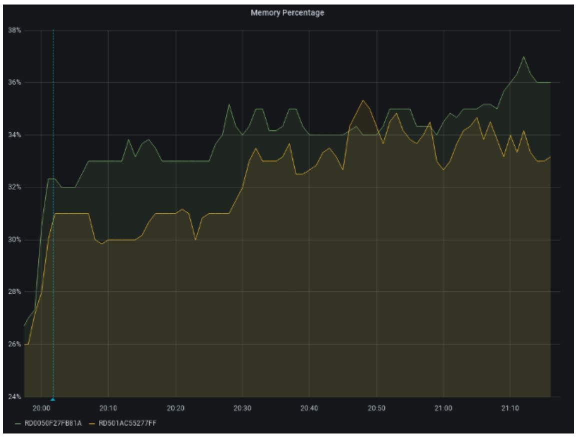
CM



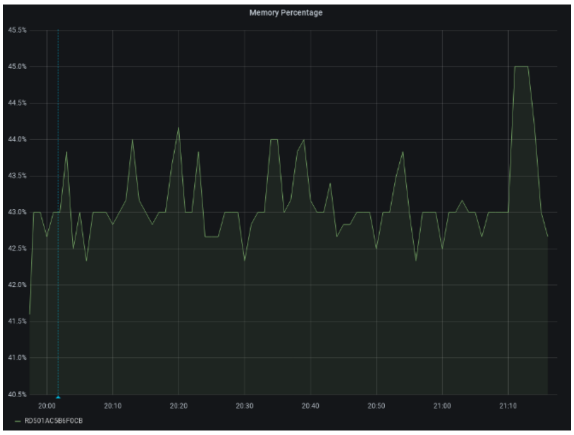
XC-Basic



XC Resource Intensive

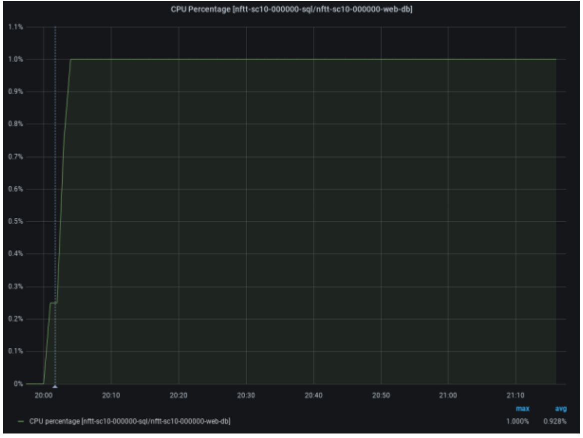


Prc

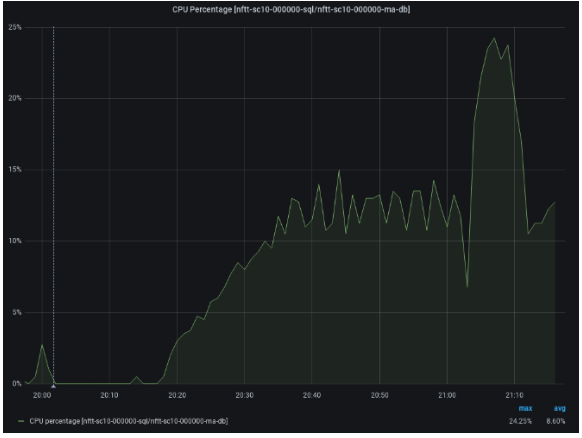


Azure SQL databases

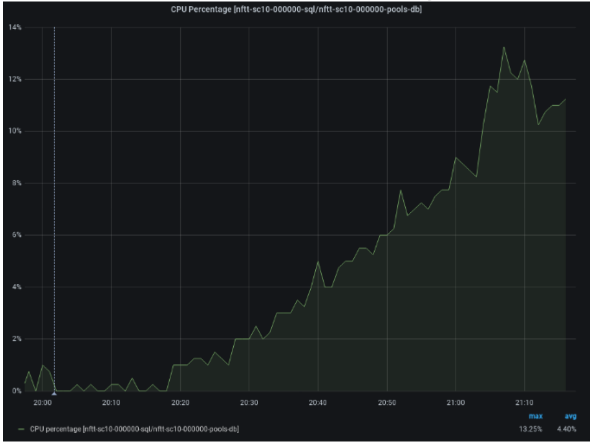
WEB DB. CPU Percentage



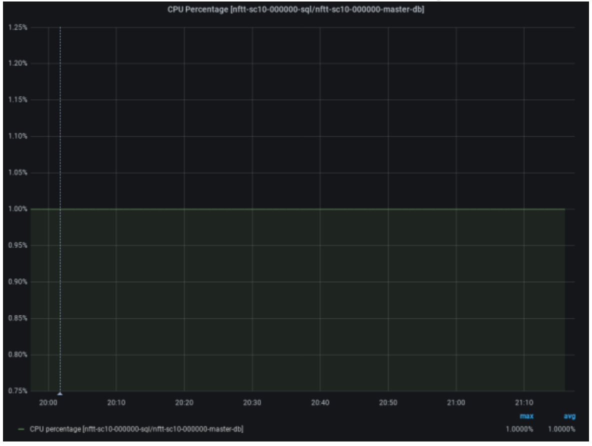
MA DB. CPU Percentage



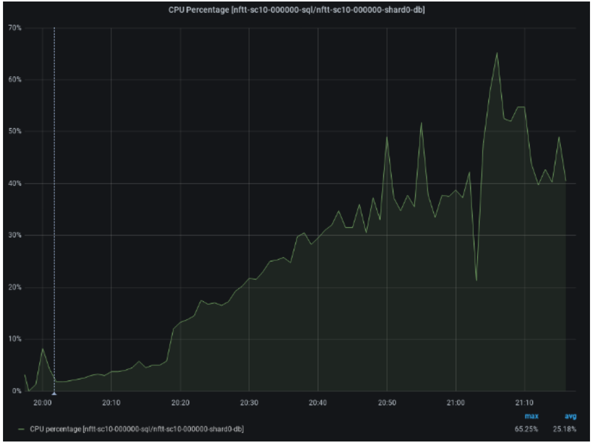
Pools DB. CPU Percentage



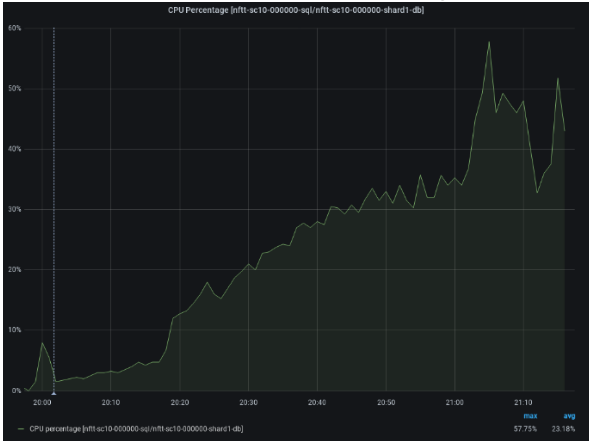
Master DB. CPU Percentage



Shard-0 DB. CPU Percentage

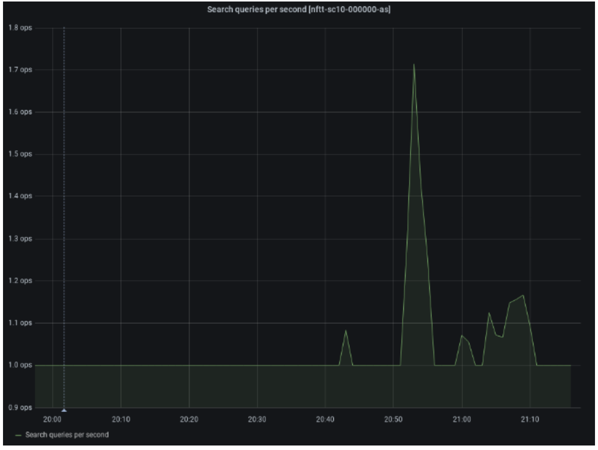


Shard-1 DB. CPU Percentage

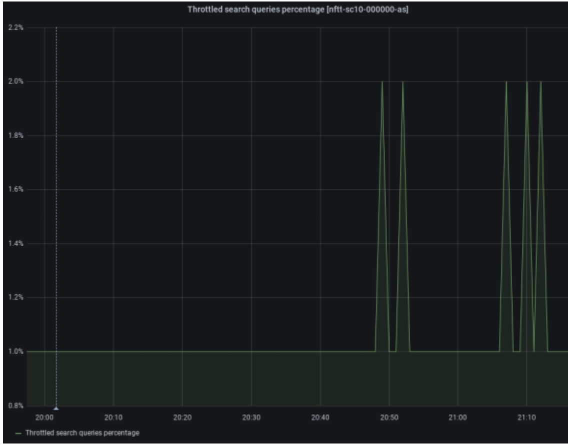


Azure Search

Search queries per sec

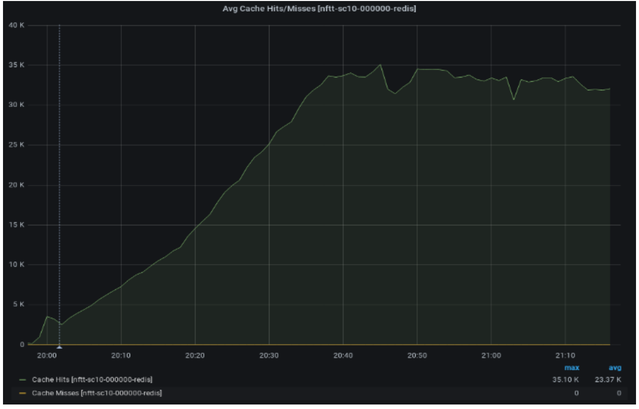


Throttled search queries percentage

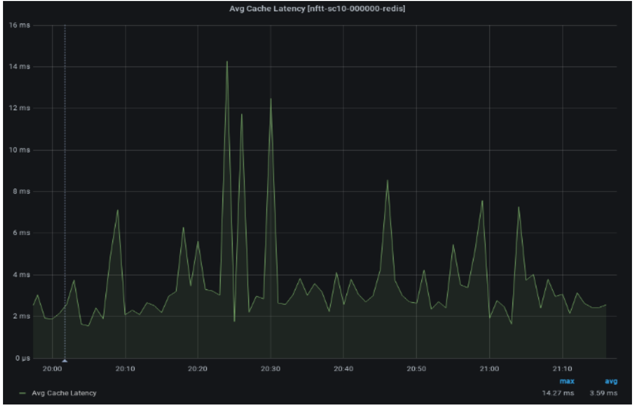


Azure Cache - Redis

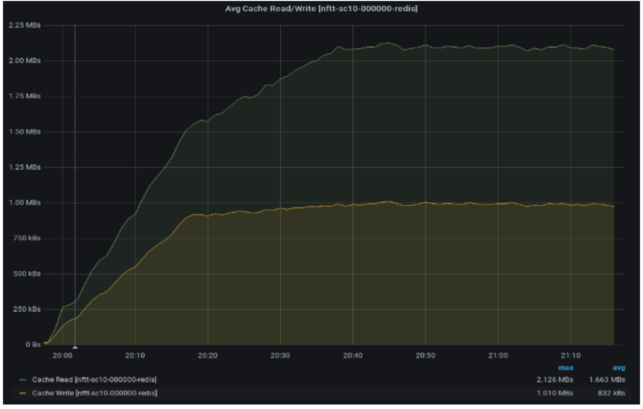
Avg Cache Hits&Misses



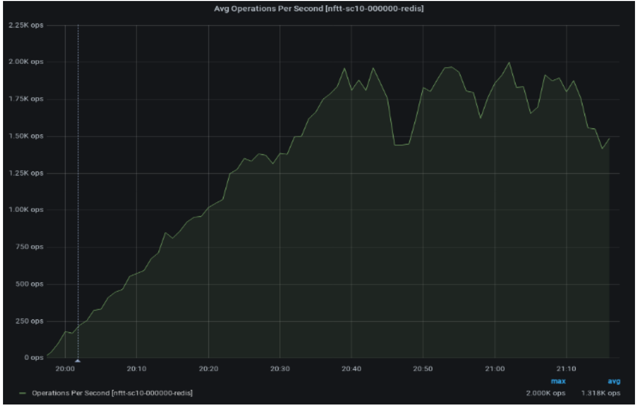
Avg Cache Latency



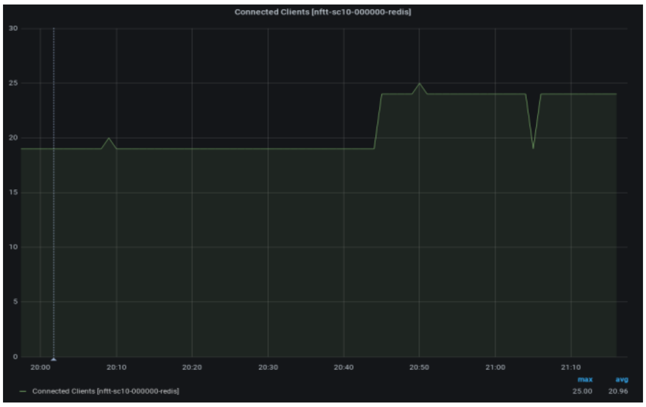
Avg Cache Read&Write



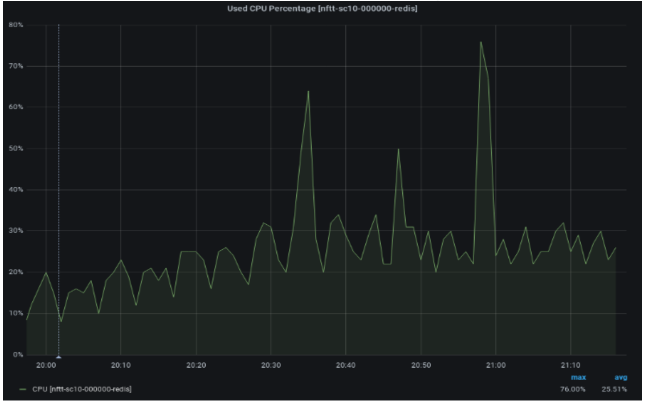
Avg Operations Per Second



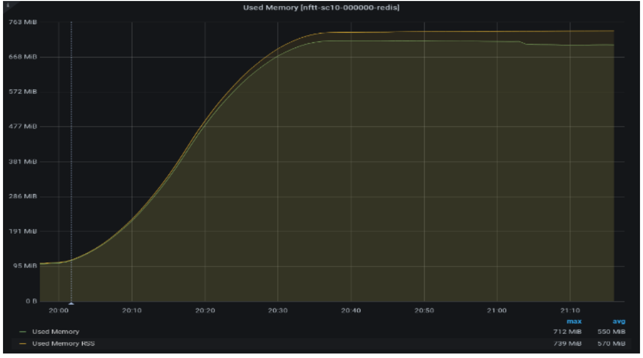
Connected Clients



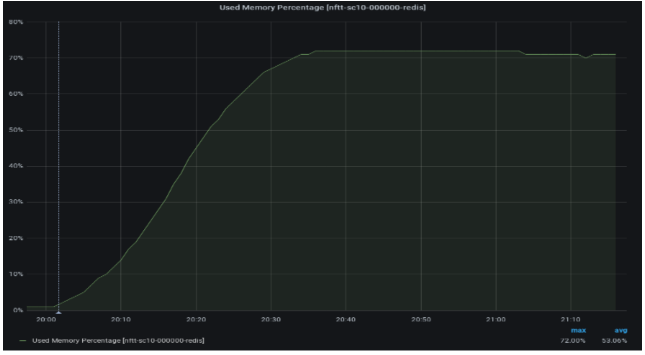
Used CPU Percentage



Used Memory

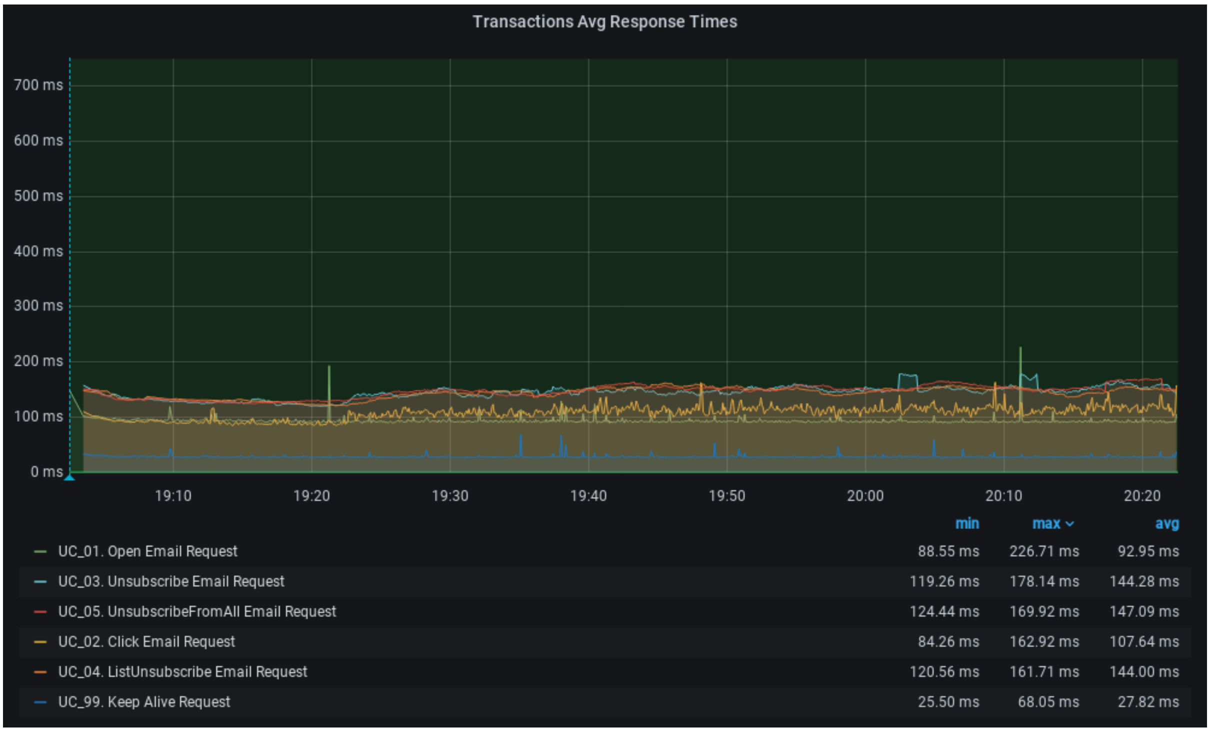


Used Memory Percentage



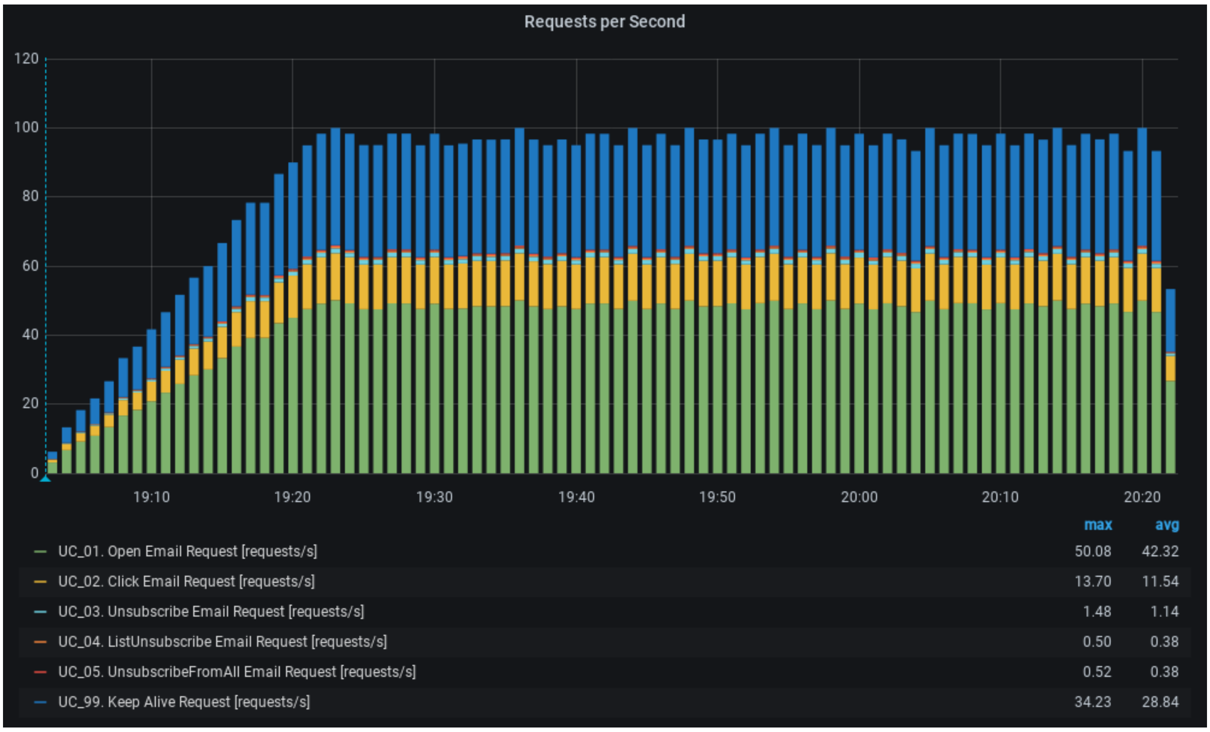
Personalized email campaign

Response times



	min	max	avg
UC_01. Open Email Request	88.55 ms	226.71 ms	92.95 ms
UC_03. Unsubscribe Email Request	119.26 ms	178.14 ms	144.28 ms
UC_05. UnsubscribeFromAll Email Request	124.44 ms	169.92 ms	147.09 ms
UC_02. Click Email Request	84.26 ms	162.92 ms	107.64 ms
UC_04. ListUnsubscribe Email Request	120.56 ms	161.71 ms	144.00 ms
UC_99. Keep Alive Request	25.50 ms	68.05 ms	27.82 ms

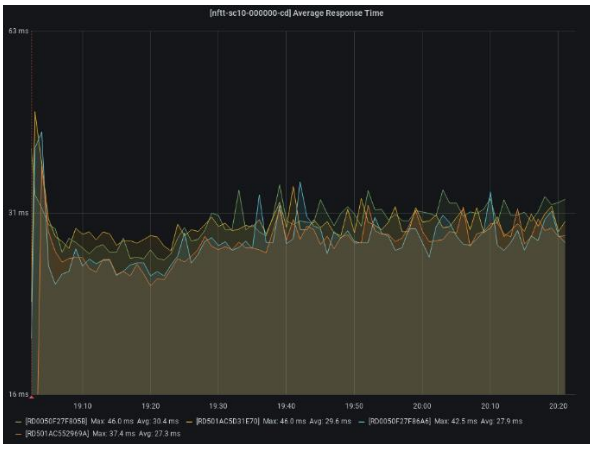
Requests per second



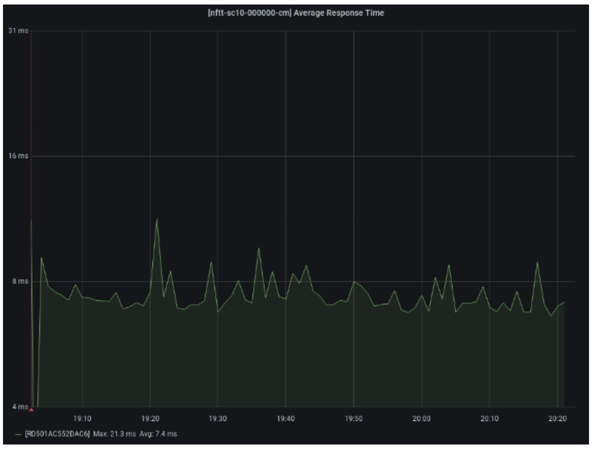
	max	avg
UC_01. Open Email Request [requests/s]	50.08	42.32
UC_02. Click Email Request [requests/s]	13.70	11.54
UC_03. Unsubscribe Email Request [requests/s]	1.48	1.14
UC_04. ListUnsubscribe Email Request [requests/s]	0.50	0.38
UC_05. UnsubscribeFromAll Email Request [requests/s]	0.52	0.38
UC_99. Keep Alive Request [requests/s]	34.23	28.84

Response times – app service plan

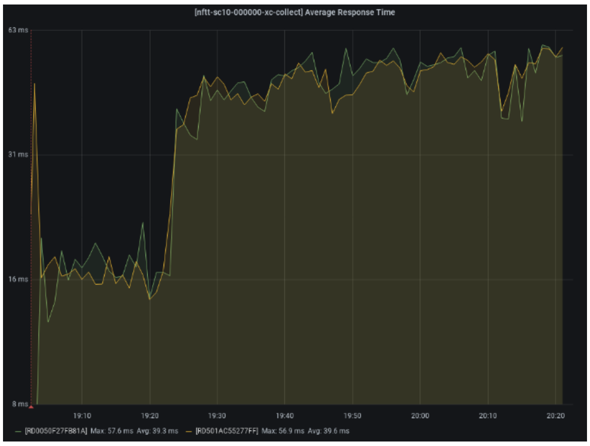
CD



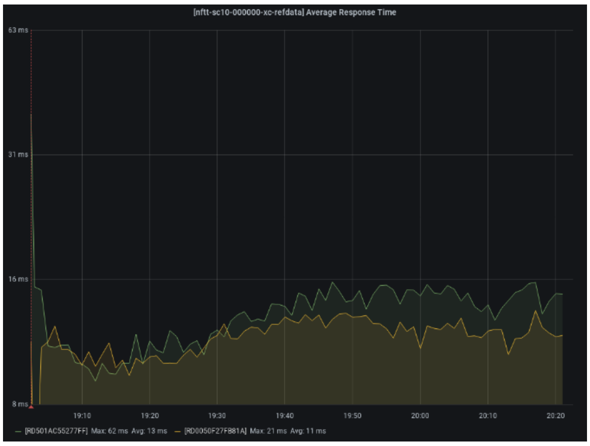
CM



XC-collection

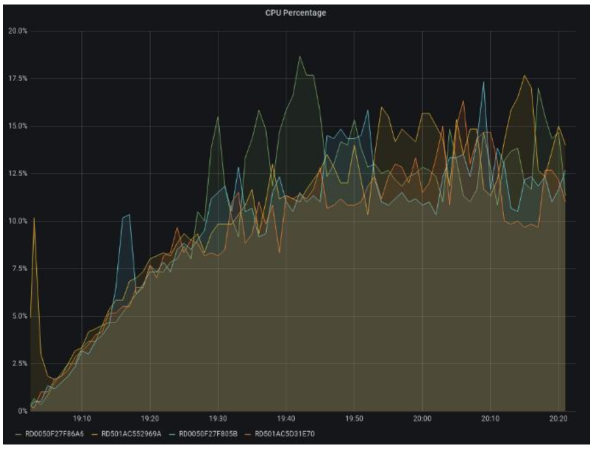


RefData

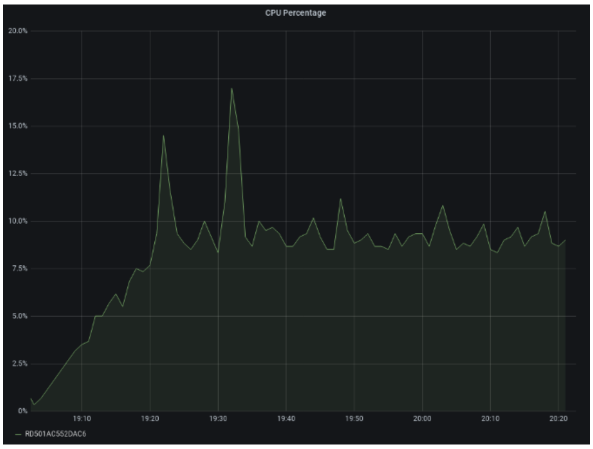


CPU – app service plan

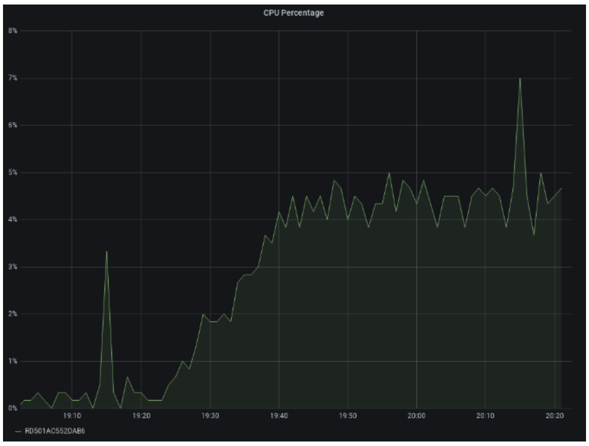
CD



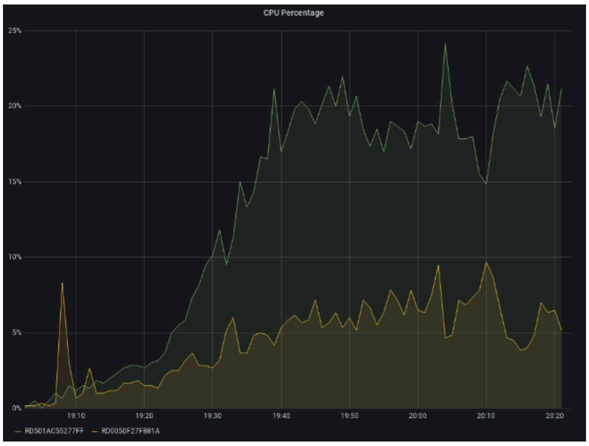
CM



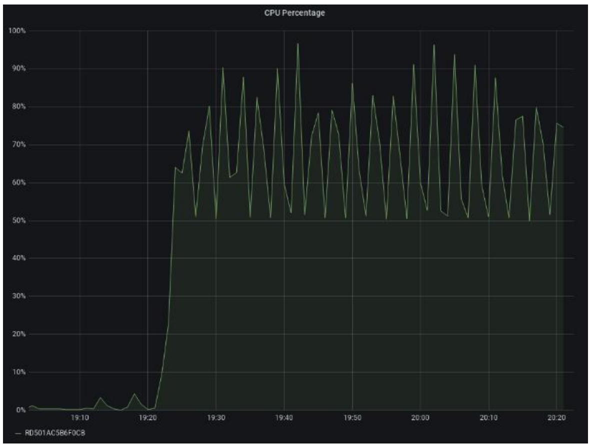
XC-Basic



XC Resource Intensive

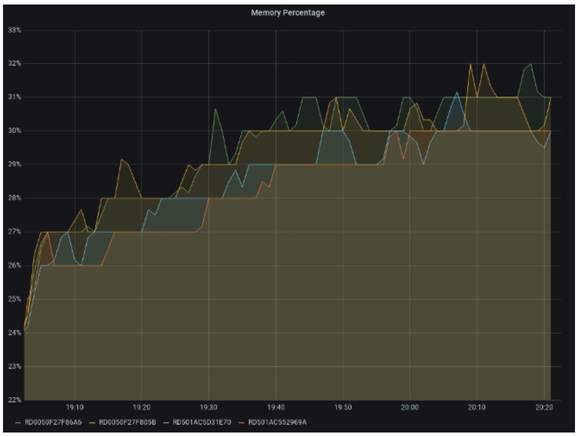


Prc

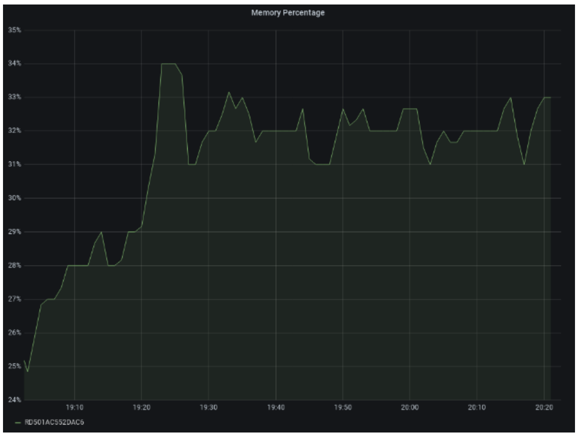


Memory – app service plan

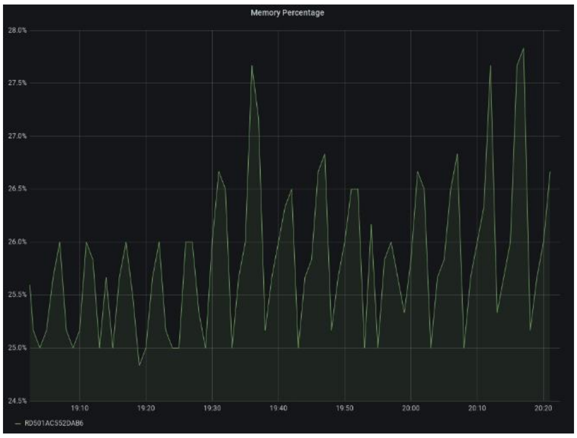
CD



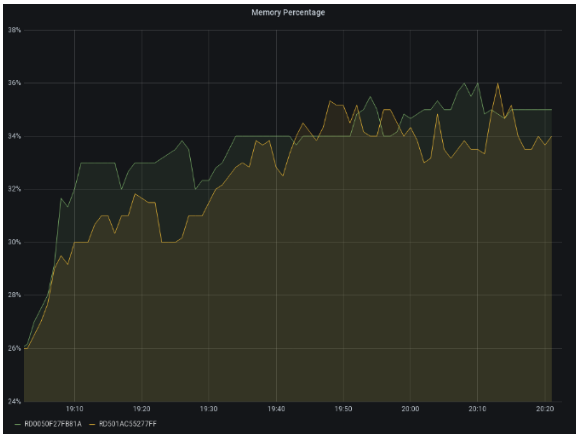
CM



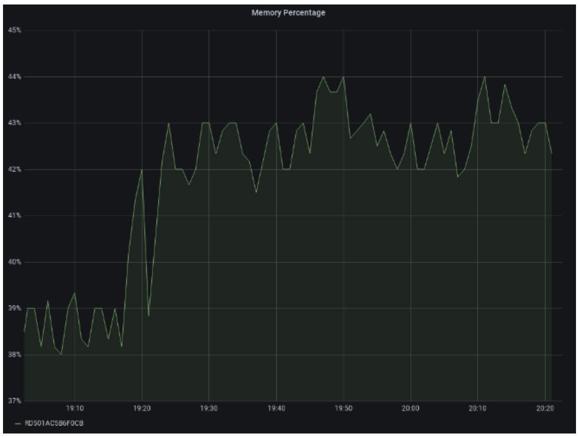
XC-Basic



XC Resource Intensive

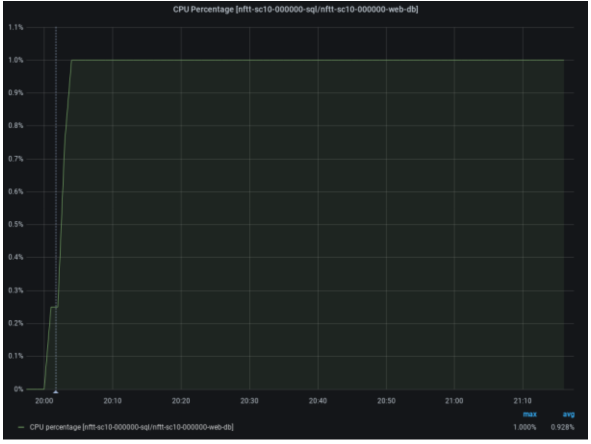


Prc

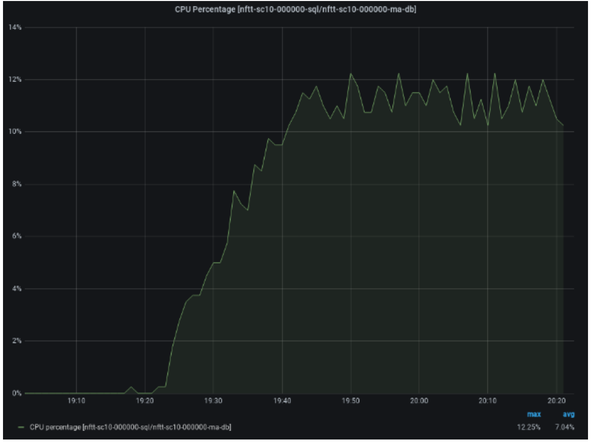


Azure SQL databases.

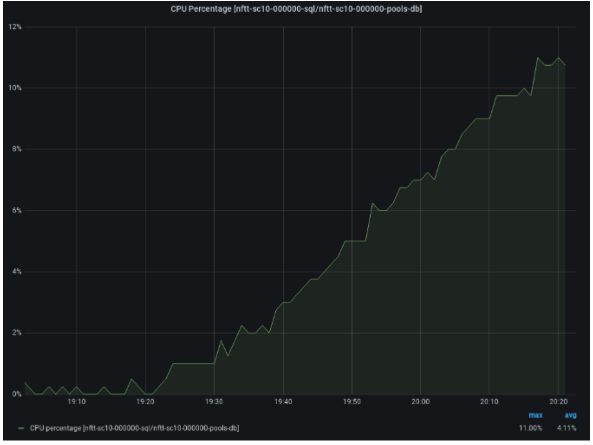
WEB DB. CPU Percentage



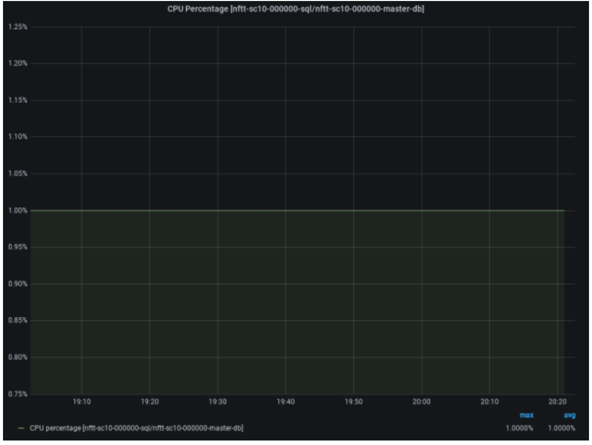
MA DB. CPU Percentage



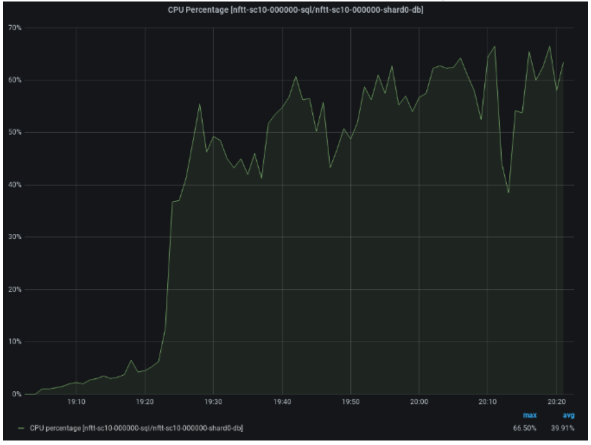
Pools DB. CPU Percentage



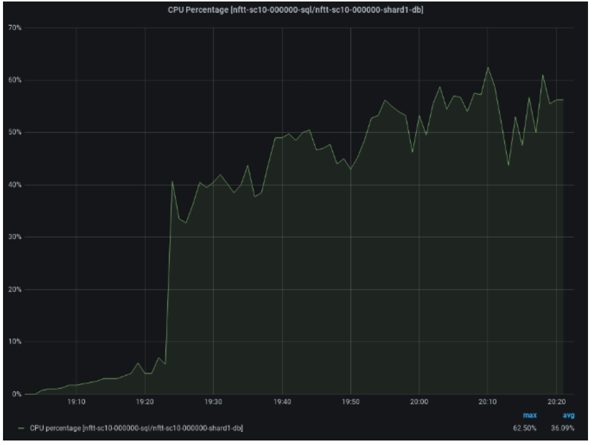
Master DB. CPU Percentage



Shard-0 DB. CPU Percentage

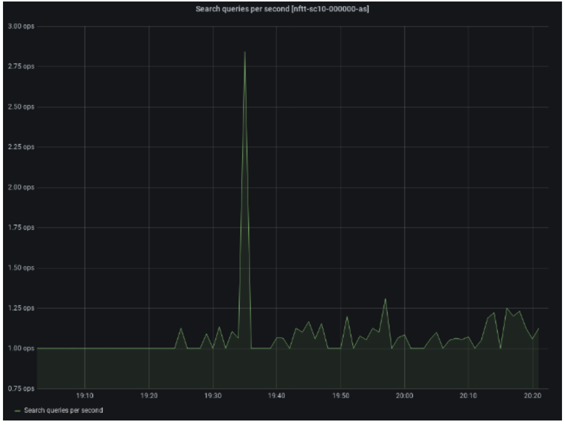


Shard-1 DB. CPU Percentage

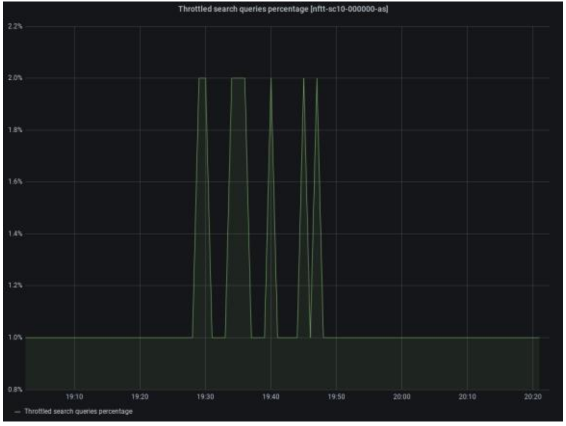


Azure Search

Search queries per sec

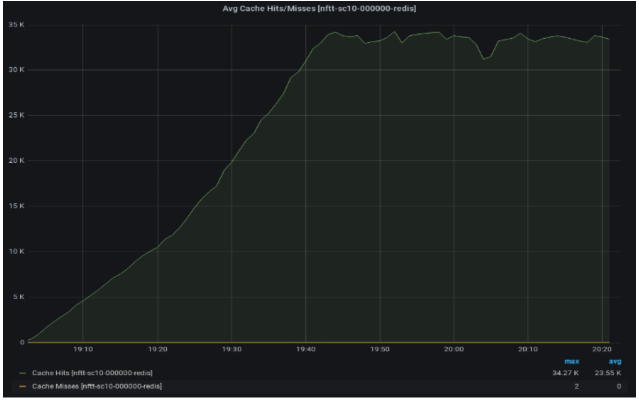


Throttled search queries percentage

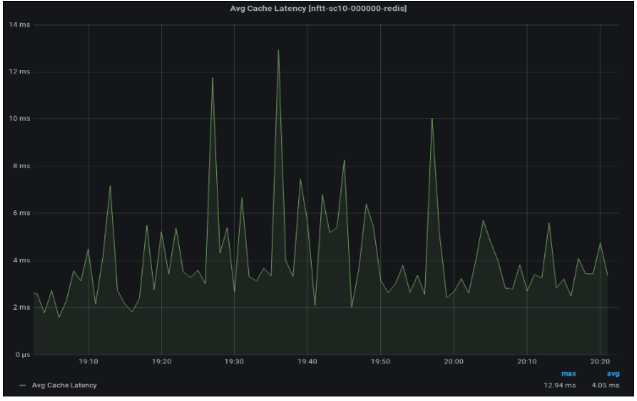


Azure Cache - Redis

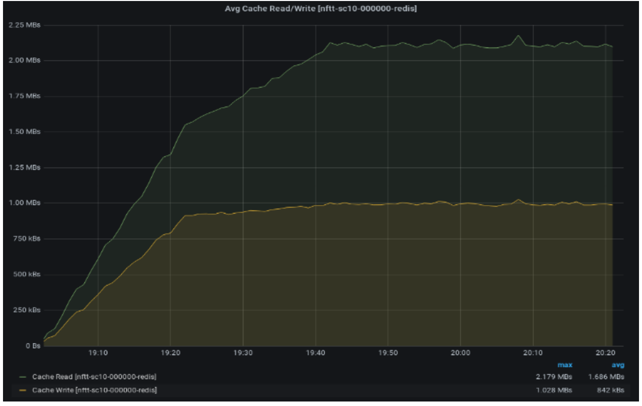
Avg Cache Hits&Misses



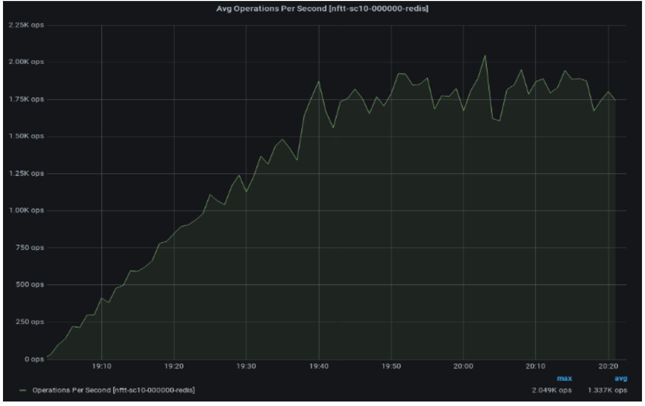
Avg Cache Latency



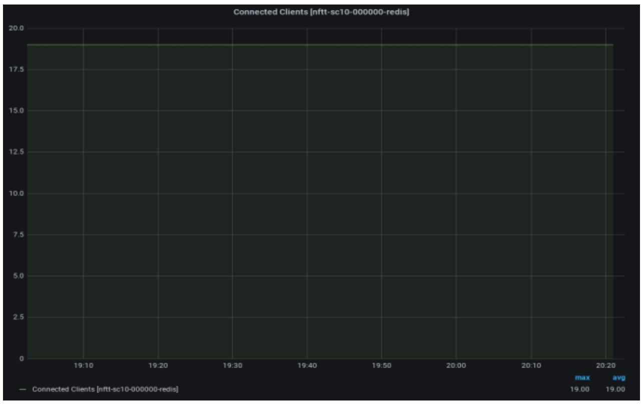
Avg Cache Read&Write



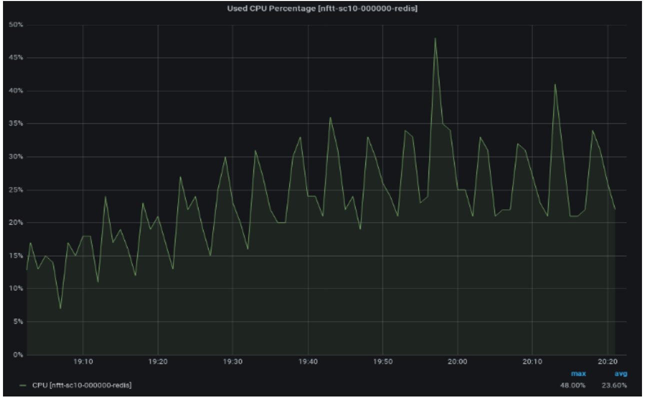
Avg Operations Per Second



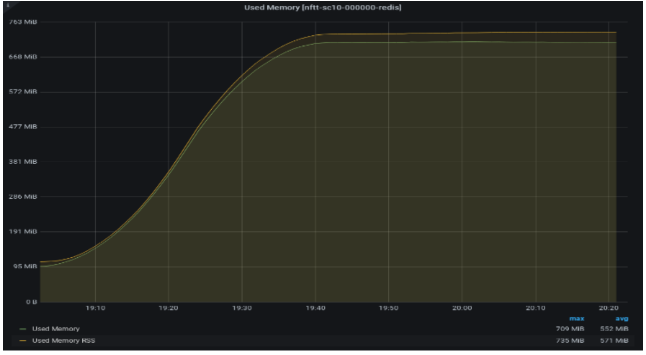
Connected Clients



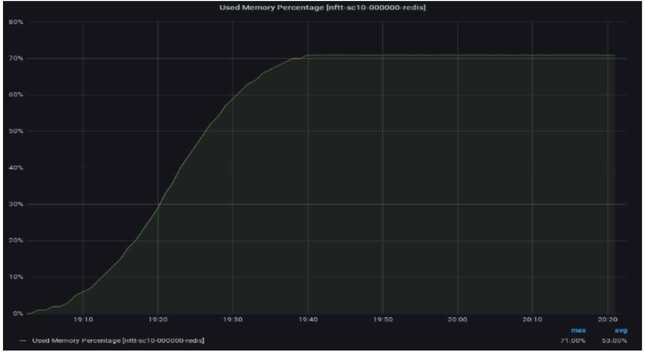
Used CPU Percentage



Used Memory



Used Memory Percentage


Used Memory Percentage [nftt-sc10-000000-redis]
19:10
19:20
19:30
19:40
19:50
20:00
20:10
20:20
max
avg
Used Memory Percentage [nftt-sc10-000000-redis]