Azure服务总线性能基准测试示例代码

本文提供对Azure服务总线进行基准测试的示例代码。主要通过设置Prefetch和MaxConcurrentCalls来加快消息的获取和处理。更多性能优化建议参阅[这里](https://www.azure.cn/documentation/articles/service-bus-performance-improvements/)。

## 发送端

生成消息，并分批，然后按批次异步发送。

|  |  |
| --- | --- |
| // Generate message batches  var batchMsgDic = new Dictionary<int, List<BrokeredMessage>>();  var batchMsgs = new List<BrokeredMessage>(batchSize);  var batchCount = 0;  for (int i = 1; i <= messagesCount; i++)  {  // Create message, passing a string message for the body.  var message = new BrokeredMessage($"Test message {i}");  // Set additional custom app-specific property.  message.Properties["MessageId"] = Guid.NewGuid();  message.Properties["CreateTime"] = DateTime.UtcNow.ToString("HH:mm:ss.fff", CultureInfo.InvariantCulture);  Console.WriteLine($"{DateTime.UtcNow.ToString("HH:mm:ss.fff", CultureInfo.InvariantCulture)} --- Create message {i}");  batchMsgs.Add(message);  if (i % batchSize == 0)  {  batchCount++;  batchMsgDic.Add(batchCount, batchMsgs);  batchMsgs = new List<BrokeredMessage>(batchSize);  }  }  // Send message batches asynchronoursly without waiting  var sendTasks = new List<Task>(batchCount);  foreach (var batch in batchMsgDic)  {  sendTasks.Add(queueClient.SendBatchAsync(batch.Value));  Console.WriteLine($"{DateTime.UtcNow.ToString("HH:mm:ss.fff", CultureInfo.InvariantCulture)} --- Sent batch {batch.Key}");  }  Task.WaitAll(sendTasks.ToArray());  Console.WriteLine("All messages are sent!"); |  |

## 接收端

设定Prefetch使得从服务总线队列中获取消息时，一次性可以获取设定条数的消息，我们这里设定1000就相当于一次性获取所有消息。

设定MaxConcurrentCalls使得同时可以有设定数目的线程数来处理消息，我们这里设定1000就相当于最多有1000个线程来处理消息，从而保证消息的最快处理。

|  |
| --- |
| // TODO: update the name of your queue  const string QueueName = "yourqueuename";  const int PrefetchCount = 1000;  const int MaxThreadsCount = 1000;  // QueueClient is thread-safe. Recommended that you cache  // rather than recreating it on every request  QueueClient Client;  ManualResetEvent CompletedEvent = new ManualResetEvent(false);  public override void Run()  {  Trace.WriteLine("Starting processing of messages");  // Initiates the message pump and callback is invoked for each message that is received, calling close on the client will stop the pump.  Client.OnMessageAsync(async (receivedMessage) =>  {  Trace.WriteLine($"Rcv Msg {receivedMessage.MessageId} --- C:{receivedMessage.Properties["CreateTime"]} | E:{receivedMessage.EnqueuedTimeUtc.ToString("HH:mm:ss.fff", CultureInfo.InvariantCulture)} | R:{DateTime.UtcNow.ToString("HH:mm:ss.fff", CultureInfo.InvariantCulture)}");  // sleep 1s to simulate processing  await Task.Delay(1000);  // Process the message  Trace.WriteLine($"End Msg {receivedMessage.MessageId} --- F:{DateTime.UtcNow.ToString("HH:mm:ss.fff", CultureInfo.InvariantCulture)}");  }, new OnMessageOptions { AutoComplete = true, MaxConcurrentCalls = MaxThreadsCount });  CompletedEvent.WaitOne();  }  public override bool OnStart()  {  // Set the maximum number of concurrent connections  ServicePointManager.DefaultConnectionLimit = 12;  // Create the queue if it does not exist already  string connectionString = CloudConfigurationManager.GetSetting("Microsoft.ServiceBus.ConnectionString");  var namespaceManager = NamespaceManager.CreateFromConnectionString(connectionString);  if (!namespaceManager.QueueExists(QueueName))  {  namespaceManager.CreateQueue(QueueName);  }  // Initialize the connection to Service Bus Queue  Client = QueueClient.CreateFromConnectionString(connectionString, QueueName);  Client.PrefetchCount = PrefetchCount;  return base.OnStart();  } |

## 运行

1. 设置服务总线信息

发送端：

|  |
| --- |
| // TODO: update with your own value here  var sbConnStr = "yourservicebusconnectionstring";  var queueName = "yourqueuename"; |

接收端：

|  |
| --- |
| // TODO: update the name of your queue  const string QueueName = "yourqueuename"; |

|  |
| --- |
| <ConfigurationSettings>  <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" value="DefaultEndpointsProtocol=https;AccountName=[name];AccountKey=[key];EndpointSuffix=core.chinacloudapi.cn" />  <Setting name="Microsoft.ServiceBus.ConnectionString" value="Endpoint=sb://[your namespace].servicebus.windows.net;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=[your key]" />  </ConfigurationSettings> |

1. 发布接收端项目ReceiverWorkerRole到Azure上
2. 运行发送端来发送消息
3. 查看步骤1中设置的存储账户中的WADLogsTable的日志，可通过[Microsoft Storage Explorer](https://na01.safelinks.protection.outlook.com/?url=http%3A%2F%2Fstorageexplorer.com%2F&data=02%7C01%7Cxingzhou%40microsoft.com%7Cc2517b60734a4c84a03908d4870b3bf0%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C636281929741366036&sdata=4PjFoYMZGNWRq6BZMaDbcqve6opiAdc8XnrMh9z1zAw%3D&reserved=0)来查看。

日志分析：

* Rcv Msg 6ca4a6f26dcd4aaf956e422c90e5aee5 --- **C**:06:42:17.934 | **E**:06:42:19.192 | **R**:06:42:19.61
  + C代表消息创建时间
  + E代表消息到达服务总线队列时间
  + R代表消息接收到的时间
* End Msg 6ca4a6f26dcd4aaf956e422c90e5aee5 --- **F**:06:42:20.652
  + F代表消息处理完成时间

源代码

<https://github.com/allenhula/azure-china-get-started/tree/master/ServiceBus/CSharp/Benchmark%20Testing>