Re: HttpListener BeginGetContext does not seem to handle more than 2 request simultaneously

Source:
http://www.tech−archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2006−12/msg00559.html

- *From*: "Kunal" <koolkunal@xxxxxxxxx>
- *Date*: 29 Dec 2006 12:53:40 −0800

Hi Vadym,

After the change you suggested, my server ran just fine and seemed to
be aceepting/processing multiple requests simultaneously. However,
after a few hours of operation (when there are about 15 clients coming
in per second) the
server crashed with the followin exception −

System.OutOfMemoryException: Exception of type
'System.OutOfMemoryException' was thrown.

Server stack trace:
at System.Net.RequestContextBase.SetBuffer(Int32 size)
at System.Net.AsyncRequestContext.Allocate(UInt32 size)
at System.Net.HttpListener.BeginGetContext(AsyncCallback callback,
Object state)
at HttpListenerLibrary.HttpListenerWrapper.ProcessRequest() in
C:\StarTeam\easycare\Software\Release\SimulatorsDev\AcsSim\HttpListenerLibrary\HttpListenerLibrary.cs:line
260
at
System.Runtime.Remoting.Messaging.StackBuilderSink._PrivateProcessMessage(IntPtr
md, Object[] args, Object server, Int32 methodPtr, Boolean
fExecuteInContext, Object[]& outArgs)
at
System.Runtime.Remoting.Messaging.StackBuilderSink.PrivateProcessMessage(RuntimeMethodHandle
md, Object[] args, Object server, Int32 methodPtr, Boolean
fExecuteInContext, Object[]& outArgs)
at
System.Runtime.Remoting.Messaging.StackBuilderSink.SyncProcessMessage(IMessage
msg, Int32 methodPtr, Boolean fExecuteInContext)

Exception rethrown at [0]:
at
System.Runtime.Remoting.Proxies.RealProxy.HandleReturnMessage(IMessage
reqMsg, IMessage retMsg)
at
System.Runtime.Remoting.Proxies.RealProxy.PrivateInvoke(MessageData&

msgData, Int32 type)
at HttpListenerLibrary.HttpListenerWrapper.ProcessRequest()
at HttpListenerLibrary.HttpListenerController.Pump() in
C:\StarTeam\easycare\Software\Release\SimulatorsDev\AcsSim\HttpListenerLibrary\HttpListenerLibrary.cs:line
166

There was this event also in the event log −

Exception information:
Exception type: OutOfMemoryException
Exception message: Exception of type 'System.OutOfMemoryException'
was thrown.

Thread information:
Thread ID: 7
Thread account name: CPEPR1\Administrator
Is impersonating: False
Stack trace: at System.Web.HttpRequest.GetEntireRawContent()
at System.Web.HttpRequest.get_InputStream()
at System.Web.Services.Protocols.SoapServerProtocol.Initialize()
at System.Web.Services.Protocols.ServerProtocolFactory.Create(Type
type, HttpContext context, HttpRequest request, HttpResponse response,
Boolean& abortProcessing)
at
System.Web.Services.Protocols.WebServiceHandlerFactory.CoreGetHandler(Type
type, HttpContext context, HttpRequest request, HttpResponse response)
at
System.Web.Services.Protocols.WebServiceHandlerFactory.GetHandler(HttpContext
context, String verb, String url, String filePath)
at System.Web.HttpApplication.MapHttpHandler(HttpContext context,
String requestType, VirtualPath path, String pathTranslated, Boolean
useAppConfig)
at
System.Web.HttpApplication.MapHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute()
at System.Web.HttpApplication.ExecuteStep(IExecutionStep step,
Boolean& completedSynchronously)

Do you think the server is not getting enough memory to process further
requests ? from the exception it looks like while trying to allocate
buffers for storing the incoming request it fails. But this is within
the stack area. How can this be debugged ? How can I find out what are
the resources occupying memory at that time ?

Thanks,
Kunal

On Dec 26, 10:55 pm, "Kunal" <koolku...@xxxxxxxxx> wrote:

Hi Vadym,

Thanks for your suggestion. This change seems to work for me.
However, what is the best way of determining how many concurrent
connections
my server is handling at any instant ? And how to make sure any
connections
are not being refused or dropped ?

In my code, there's one change I tried but that threw an exception.
Instead of
calling ProcessRequest in the infinite while loop, I call it just once.
Then within
the ListenerCallback, I setup a new BeginGetContext just after the call
to
EndGetContext, to set up a loop. But I got an exception related to a
thread
exit at the time of call.

Do you think the current while loop is just fine and I need not try out
this new
way ? I saw this implementation technique somewhere and was curious to
try
it out.

Thanks for all your time,

Regards,
Kunal

On Dec 22, 1:54 pm, "Vadym Stetsyak" <vady...@xxxxxxx> wrote:

> Hello, Kunal!
>
> line "result.AsyncWaitHandle.WaitOne();"
> blocks in ProcessRequest method thus breaking the async execution.
>
> Try to comment that line an repeat your tests.
>
> You wrote on 21 Dec 2006 14:41:15 −0800:

K> Hi Friends,

K> I'm trying to host a webservice that will receive/process multiple
K> client requests simultaneously. For this purpose, I wrote the
K> following
K> code, but it does not seem to be handling more than two at a time. I
K> put in a few console prints and have also attached the output. Here
K> it
K> goes −

K> public void serverStart() {

K> HttpListener _listener = new HttpListener();

K> _listener.Prefixes.Add(.....);

K> _listener.Start();

K> while (true)

K> ProcessRequest();

K> }

K> public void ProcessRequest()

K> {

K> Console.WriteLine("PR " + ++count); //count initialized to 0 in
K> beginning

K> IAsyncResult result = _listener.BeginGetContext(new AsyncCallback
K> ListenerCallback),this._listener);

K> result.AsyncWaitHandle.WaitOne();

K> }

K> protected void ListenerCallback(IAsyncResult result)

K> {

K> if (this._listener == null) return;

K> HttpListenerContext context =
K> this._listener.EndGetContext(result);

K> Console.WriteLine("LC " + count);

K> this.ProcessRequest2(context);

K> }

K> public void ProcessRequest2(HttpListenerContext ctx)

K> {

K> Console.WriteLine("PR2 " + count);

K> string str = ctx.Request.HttpMethod;

K> HttpListenerWorkerRequest workerRequest =

K> new HttpListenerWorkerRequest(ctx,_virtualDir, _physicalDir,
K> _logCallback, _pxeb);

K> HttpRuntime.ProcessRequest(workerRequest);

K> }

K> The output this gives me looks like this −

K> PR 1
K> PR 2
K> LC 2
K> PR2 2
K> LC 2
K> PR2 3
K> PR 3
K> LC 3
K> PR2 4
K> PR 4
K> LC 4
K> PR2 5
K> PR 5
K> PR 6
K> LC 6
K> PR2 6
K> LC 6
K> PR2 7

K> PR 7
K> PR 8
K> LC 8
K> PR2 8
K> PR 9
K> LC 9
K> PR2 9
K> LC 9

K> Can someone point out what's happening ? The Microsoft documentation
K> does not say anything about the no. of connections this model can
K> handle or has a limit to. Any developers who have used this may
K> please
K> enlighten me !

K> Thanks and Regards,

K> Kunal

With best regards, Vadym Stetsyak.
Blog:http://vadmyst.blogspot.com− Hide quoted text −− Show quoted text −

.