

Re: aspnet_state.exe Internals Info Needed

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.aspnet/2005-02/6525.html>

From: MattC (*m_at_m.com*)

Date: 02/22/05

Date: Tue, 22 Feb 2005 09:43:47 -0000

Karl,

Last night I tried runing with server side viewstate off, indeed the server was fine, the processor was much less under pressure, and the memory usage was constant. Which means there must be a leak in my viewstate persistence code (below).

My problem is that I have a timesheet data entry page for the application. Users can make entries for each day, selecting an activity, each activity requires a different set of boxes, this requires a postback. So they may have 50/60 postbacks per session. All being persisted to the session.

```
private string NewViewState
{
    get{ return "VIEWSTATE_"; }
}

private string ViewStateKey
{
    get{ return "__VIEWSTATE_KEY"; }
}

private string UniqueViewStateKey
{
    get{ return NewViewState + Request.UserHostAddress + "_" +
Guid.NewGuid().ToString(); }
}

protected override void SavePageStateToPersistenceMedium(object viewState)
{
    try
    {
        if(Conquest.ConfigurationData.ViewStateStorage ==
ViewStateStorage.Session)//if we are implementing our on server viewstate
        {
            string str = this.UniqueViewStateKey;
            LosFormatter oLosFormatter = new LosFormatter();
            StringWriter oStringWriter = new StringWriter();
```

```
oLosFormatter.Serialize(oStringWriter, ViewState); //serialize ViewState
into string writer
oLosFormatter = null;

Session[str] = oStringWriter.ToString(); //store in state server
oStringWriter = null;

RegisterHiddenField(ViewStateKey, str);
RegisterHiddenField("__VIEWSTATE", String.Empty);
}
else
{
base.SavePageStateToPersistenceMedium(ViewState);
}
}
catch(Exception e)
{
throw new
ConquestException("PerformancePage.SavePageStateToPersistenceMedium Failed",
null, e);
}
}

protected override object LoadPageStateFromPersistenceMedium()
{
try
{
if(Conquest.ConfigurationData.ViewStateStorage ==
ViewStateStorage.Session)//if we are implementing our on server ViewState
{
object ViewState = null; //return ViewState
string str = Request.Form[ViewStateKey];
LosFormatter oLosFormatter = new LosFormatter();

if (!str.StartsWith(NewViewState)) //check the ViewState exists
{
Conquest.Events.WriteToLog("Failed to locate VIEWSTATE KEY: " + str
+ "");
ViewStatePersistenceFailure(str); //failure
}
else //view state does exist
{
try
{
string tmp = Session[str].ToString(); //get the serialize ViewState
from session
ViewState = oLosFormatter.Deserialize(tmp); //deserialize into object
oLosFormatter = null;
}
catch(Exception e) //catch all and handle gracefully
{
```

microsoft.public.dotnet.framework.aspnet: Re: aspnet_state.exe Internals Info Needed

```
Conquest.Events.WriteToLog("Failed to Load ViewState from " +
Conquest.ConfigurationData.ViewStateStorage.ToString() +
" " +
ConquestException.HandleConquestException(e),System.Diagnostics.EventLogEntryType.Information);

ViewStatePersistenceFailure(str);//Response.Redirect to homepage
}
}
//return what we got from deserialized viewstate
return viewstate;
}
else //use standard viewstate
{
return base.LoadPageStateFromPersistenceMedium();
}
}
catch(Exception e)
{
throw new
ConquestException("PerformancePage.LoadPageStateFromPersistenceMedium
failed", null, e);
}
}
TIA
```

MattC

"Karl Seguin" <karl REMOVE @ REMOVE openmymind REMOVEMETOO . ANDME net>
wrote in message news:%238F4dHEGFHA.2296@TK2MSFTNGP15.phx.gbl...

> *Hey Matt,*
> *Check out <http://www.thecodeproject.com/aspnet/ASPNETSessionInternals.asp>*
> *it's just a starter...and there's no "part 2" as the author promises.*
> *Most*
> *of my research came from digging into the .net code with Reflector and*
> *getting an understanding of the internals.*
>
> *My initial suspicion is still that the GC simply isn't running, but it's*
> *possible that you do ahve a memory leak (managed or not)...you can check*
> *out*
> *<http://blogs.msdn.com/akhune/archive/2004/06/11/153734.aspx> as a starting*
> *point for this.*
>
> *Also, I wouldn't/couldn't say that those are the reasons your CPU is*
> *maxing*
> *out (certainly if you are looping 1million times per page hit, that might*
> *be*
> *worth investigating too ;)). I've certainly seen HTTP Compression take*
> *a*
> *chunk out of CPU cycles...so simply try turning it off and see if it's any*
> *better? Similarly, simply try to turn off the server-side viewstate and*
> *observe the effect. Your box certainly is old and I would simply suspect*
> *that it'll be able to handle a certain load and no more...if you are*

Re: aspnet_state.exe Internals Info Needed

> trying
> to run an application where you feel the need to simulate 1million session
> entries, you likely need more hardware...
>
> If the basic stuff doesn't work (although I'm hoping turning off
> HttpCompression for testing will ease the cpu a lot), get a profiler and
> take a spin in your code. Ant's profiler is my preferred tool, and comes
> with a full 14 day trial:
> http://www.red-gate.com/code_profiling.htm
>
> Karl
>
> --
> MY ASP.Net tutorials
> <http://www.openmymind.net/>
>
>
> "MattC" <m@m.com> wrote in message
> news:O8yTBqDGFHA.3244@TK2MSFTNGP15.phx.gbl...
>> Karl,
>>
>> Excellent, thanks for your response.
>>
>> 1.) Could you post some of the links for the articles you found.
>>
>> 2.) I am having to persist my viewstate into the session, overloading the
>> SavePageStateToPersistenceMedium and
>> LoadPageStateToPersistenceMedium methods. Even after many many requests
> the
>> W3WP process doesn't seem to go down and similarly my aspnet_state.exe
> doesn't
>> seem to rise.
>>
>> Is there any way I can call to have a particular session insertion
>> serialized immediately?
>>
>> I suspect that as my CPU is now working to its maximum due to the fact
> that
>> it is having to do HTTP compression and serialize/deserialize plus search
>> for session data it is having trouble finding the time to a.) Serialize
>> to
>> the state server, b.) Garbage collect the temporary memory allocated in
> the
>> worker process.
>>
>> My server is an old PIII 730 w/256MB Ram, W2K3 Server, IIS6.
>>
>> Would this fit your theory?
>>
>> Finally, can I create a method that is wired to the requesthandlers
>> complete event?

>>
>> TIA
>>
>> MattC
>>
>>
>> "Karl Seguin" <karl REMOVE @ REMOVE openmymind REMOVEMETOO . ANDME net>
>> wrote in message news:OY4p2eDGFHA.228@TK2MSFTNGP15.phx.gbl...
>> > Matt:
>> > Thanks for asking, I learnt a lot by trying to find the answer. I
>> > strongly
>> > suspected that objects didn't immediatly stream to the state server
>> > when
>> > added – but instead were held in memory and only streamed to the state
>> > servrer at a certain point. After doing some digging, I found out this
>> > is
>> > what's happening. Here's a breakdown:
>> >
>> > –Session state is managed by an HttpModule called SessionStateModule
>> > –Items are only truly added to their stores during the
>> > ReleaseRequestState
>> > event, which as per msdn "Occurs after ASP.NET finishes executing all
>> > request handlers. This event causes state modules to save the current
>> > state
>> > data"
>> > –Depending on which type of storage you use, different things happen.
>> > In
>> > the case of state server, the session information is serilized into
>> > memory
>> > (asp.net) and then some stuff taht confused me happened (presumably the
>> > actual storage to the state server)
>> >
>> >
>> > So, basically all those objects are stored in memory than serialized
>> > (in
>> > memory) and sent to state server. As you can see, a lot is happening
>> > in
>> > the
>> > asp.net memory (w3wp.exe). However, you'd think that after the page is
>> > done
>> > loading that memory consumption would go back down. But it probably
>> > doesn't. This is because the garbage collector hasn't passed yet.
>> > Remember, memory won't be reclaimed until its needed. So if you lower
>> > your
>> > loop so that it can finish, you'll see the memory go up and stay up.
>> > The
>> > memory goes up because initially all the items are help in–memory, it
>> > stays
>> > up because the GC hasn't passed yet.
>> >
>> > If you change ur code to:

```
>>> for (int i = 0; i < 800000; i++)
>>> {
>>> string index = "index" + i.ToString();
>>> Session[index] = "This is a Test";
>>> }
>>> GC.Collect();
>>> GC.WaitForPendingFinalizers();
>>> GC.Collect();
>>> GC.WaitForPendingFinalizers();
>>> GC.Collect();
>>> GC.WaitForPendingFinalizers();
>>>
>>> you'll hopefully see better results (but even then it isn't
>>> guaranteed)...don't do anything for a minute or so and you should see
>>> it
>>> go
>>> back down :)
>>>
>>> Also, don't use task manager..instead load up perfmon and go to .Net
>>> Memory, select "# of bytes in heap" and select the w3wp.exe
> process...will
>>> give you a much better idea :)
>>>
>>> Hope that helped,
>>> Karl
>>> --
>>> MY ASP.Net tutorials
>>> http://www.openmymind.net/
>>>
>>>
>>> "MattC" <m@m.com> wrote in message
>>> news:uxki15CGFHA.1132@TK2MSFTNGP10.phx.gbl...
>>>> I noticed when storing large amounts of information in the StateServer
>>>> Service that this does not increase in size, the worker process itself
>>>> seems
>>>> to grow.
>>>>
>>>> I set up a test case to try and prove this, I placed a button on a web
>>>> form
>>>> that ran the foloowing code.
>>>> string temp = "index";
>>>>
>>>> for(int i = 0; i < 1000000; i++)
>>>> {
>>>> string index = temp + i.ToString();
>>>> Session[index] = "This is a Test";
>>>> }
>>>>
>>>> I notices that the aspnet_state.exe's Mem Usage and VM Size barely
>>>> grew
>>>> yet
```

>> >> *my W3WP process grew to the point that I had to stop my app (yes my
>> >> own
>> >> fault iterating 1M times).*
>> >>
>> >> *I thought the State Server actually stored the session data itself, it
>> > would
>> >> seem from my example that this is not the case and that the memory
> space
>> >> used is that of the W3WP.*
>> >>
>> >> *Does the State Server only act as a marchalling services pointing
>> >> requests
>> >> for Session info to the correct memory location? If this is the case
> how
>> >> does it persist when an ASP.NET app is recycled.*
>> >>
>> >> *Is that portion of memory allocated to storing the Session held back
>> >> while
>> >> the ASP.NET app is recycled?*
>> >>
>> >> *I've been trying to find information on how the State Server works
>> >> internally but as yet have found nothing.*
>> >>
>> >> *TIA*
>> >>
>> >> *MattC*
>> >>
>> >>
>> >>
>> >>
>> >
>> >
>>
>>
>
>