

Re: xml in plain text file on heavy load.

Source:

<http://www.tech-archive.net/Archive/Internet-Server/microsoft.public.inetserver.iis/2004-02/1505.html>

From: David Wang [Msft] (*someone_at_online.microsoft.com*)

Date: 02/12/04

Date: Wed, 11 Feb 2004 20:35:19 -0800

Ok. You have to realize that when it comes to performance, you have to think and find ALL the possible bottlenecks and address them. Having the XML file be cached is only a part of the equation, so I wouldn't get too hung-up about it. For example, even if the XML file is magically and perfectly cached in memory and all the disk reads are optimized, if you somehow stick access to it behind a heavy lock that serializes the readers, your performance would still be bad. Also, having items queued is not bad; having a constantly growing queue is bad.

I want to emphasize that IIS would never be caching that XML file on its own unless it sent it as a response. It does not make sense for a web server to cache anything else since to do that correctly, it requires application-layer logic that is beyond the scope of any general-purpose webserver. IIS caches lots of other things where it is appropriate for its use —caching a file used by another resource is not one of them.

Some effective things to cache in your case are:

1. FileSystem cache — NTFS will cache file access and contents directly to memory and is ideally suited for your situation (many readers, one occasional writer). Make sure you open the XML file for shared read (so there's no contention amongst the readers), and if you only have one writer to the XML file, performance should be fine. You are responsible for managing any of your potential lock contention so that there IS no queue to begin with.
2. Cache of your XML-based operations — if you find that the query of XML data by ASP, after it's loaded, is expensive, you should start caching results of the operations.

As for your other thoughts of stashing the XML file in a Session or Application variable — if you do this, make sure your use of Session and Application is NOT synchronizing access to your ASP pages. I've seen people stash single-threaded objects inside of Application scope and then wonder why their pages are running only one at a time and performing poorly.

Finally, you must realize that in the time it takes to execute one modest ASP page, dozens of static files can be read, processed, and cached. In your case where access to the static file is gated behind the access of an

microsoft.public.inetserver.iis: Re: xml in plain text file on heavy load.

ASP page, I doubt access to the static file will be a bottleneck unless you manage to somehow create contention on that static file.

If you care about performance, you will be profiling your pages to see where the time is spent and then optimizing that path. My suspicion is that your ASP script and ODBC/SQL will be taking the lion's share of processing time per page (that's a network latency there, which is going to dwarf any disk latency). You will probably benefit more from having cacheable ASP responses than worrying about caching XML files.

```
--
//David
IIS
This posting is provided "AS IS" with no warranties, and confers no rights.
//
"John F." <john@clearmymind.com> wrote in message
news:402a3f61$0$327$ba620e4c@news.skynet.be...
Hi David,
Well, like you are guessing, it was more the second one:
"Read by ASP page on server side, manipulated, and response sent to client".
Actually it is :
  1) Fetch client variables. (ctrl input)
  2) Read the xml-file,
  3) Take out what it needs, and take action appropriate (do some dbase
action)
  4) Sent an "OK" or "KO" response to the client.
But because this xml-file will be read by many asp pages there will be a
tremendous load on reading this 3K xml-file. So I was wondering if IIS
itself was storing this xml-file in cash so it can serve faster those asp
pages. Each asp page do some other actions on the dbase, and I'm sure that
there will be no overhead on those dbase actions (regarding my experience
with asp/odbc and SQLserver).
The real question is : physical reading the xml-file. If IIS gives this
reading-task each time to NT server, then there could be some sort of
overhead on reading this file or on passing this command to the Win2k Os.
I though this could cause reading-error because there would be a physical
action (moving the heads on the disk) that takes time, I could watch
performance monitor for this kind of problems, but still, I want to know
this in front.
I also think there gonna be a long "waiting cue" for reading this file, and
that is still crazy. So if IIS is not gonna take this caching action, is
Win2k going to do this caching action for IIS?
I could store this xml-file in a session object for each user, but then is
memory my concern, isn't it? Or is it better to store it in an application
object? What will be the impact of this? (not memory, but restarting IIS
when there is an xml update?)
It is true that "XML can be viewed as hierarchical data" but in this case
the xml-data is sooo weird that I cannot store it in a dbase (meaning
splitting it up in relational tables) because then i have to use like 100
tables, and each time when there is a change in the data, there well be new
tables added and other tables being deleted and new relations will be made.
And that is what I do not want.
Sending the xml file to the client is not advisable because then I have to
move the logic also to the client, and I'm preventing browser issues
concerning the scripting language versions vs different browser versions.
So the only thing i see is:
  a) Let NT do the caching (if it will cash it)
  b) Put it in a session variable at runtime
  c) Put it in an application variable at runtime
  d) Put the xml just in a "function" so it will be cached by IIS? (But will
```

Re: xml in plain text file on heavy load.

microsoft.public.inetserver.iis: Re: xml in plain text file on heavy load.

not be user friendly when this xml is going to be changed concerning UTF8 coding)

Are there more options or am I missing something?

Best regards,

J.F.