Subject: Re: Shared Memory in Windows
Posted by Michael Wallace on Wed, 05 Apr 2006 01:00:41 GMT
View Forum Message <> Reply to Message

The Windows pagefile is just an area of hard drive space, nothing more, nothing less. The documentation appears to indicate that your options are to either use the system pagefile or a memory mapped file on disk. While the pagefile is just "an area of hard drive space" it has probably been optimized, of course I could be wrong -- this is Windows we're talking about. ;-) The optimizations of the pagefile will make it perform better than just using a regular memory mapped file, but worse than RAM.

I don't know how the CreateFileMapping() system call works, so I can't say if it tries to do something with RAM before using the pagefile or not. But seeing as how this is your only option, go with it and see how it performs. The first rule of performance tuning is to only tune when you have to. Try it out -- you may find that the pagefile works fast enough for your needs.

-Mike

Subject: Re: Shared Memory in Windows
Posted by Marc Reinig on Wed, 05 Apr 2006 22:11:35 GMT
View Forum Message <> Reply to Message

> ... shared
> memory appears to be backed in the pagefile. Does that mean that I will
> have overhead due to disk IO? Or is the pagefile actually backed in RAM
> and only uses pagefile.sys when RAM is exhausted.

The second is correct. All user RAM can be swapped out during its life. Therefore it when it is swapped out it must exist somewhere permanent so it can be restored and brought back into RAM.

If you create a shared memory region, which is not based on an existing file, it will be backed by the swapfile when that portion of the user's memory is swapped out.

If it is based on a real file, when the memory is 'swapped out', the file itself becomes the backing store.

There is no overhead until and unless it is swapped.

Marco

_____

Marc Reinig
UCO/Lick Observatory
Laboratory for Adaptive Optics

"Robbie" <retsil@iinet.net.au> wrote in message
news:1144197125.106190.289860@v46g2000cwv.googlegroups.com.. .
>
> Just a simple performance question for shared memory in Windows...
>
> I've started aquainting myself with POSIX and SYSV shared memory in
> UNIX and I was looking for a Windows equivalent. To my dismay shared
> memory appears to be backed in the pagefile. Does that mean that I will
> have overhead due to disk IO? Or is the pagefile actually backed in RAM
> and only uses pagefile.sys when RAM is exhausted.
>
> ---

>
>> From IDLs SHMMAP documentation
>
> Under Microsoft Windows, the CreateFileMapping() system call forms the
> basis for shared memory as well as memory mapped files ... To create a
> region of anonymous mapped memory instead of a mapped file, you pass a
> special file handle (0xffffffff) to CreateFileMapping(). In this case,
> the disk space used to back the shared memory is taken from the system
> pagefile.
>
> ---

>
>> From the MSVC documentation for CreateFileMapping()
>
> .... In this case, CreateFileMapping creates a file mapping object of
> the specified size backed by the operating-system paging file rather
> than by a named file in the file system.
>
> ---

>
> Thanks,
> Robbie
>

Subject: Re: Shared Memory in Windows
Posted by Robbie on Thu, 06 Apr 2006 02:11:11 GMT

Thanks for the heads up on the Windows page file. I ran some test code to observe it using pagefile.sys when the segment was larger than RAM. It's very comforting to see the theory match the practical results.

> The first rule of performance tuning is to only tune when
> you have to. Try it out -- you may find that the pagefile works fast
> enough for your needs.

Well, I'm actually investigating shared memory so I can make a decision on my data model. I'm mid-way through my project and I have deliberately left the data model quite open. However, I can't leave it open forever or else I will start to loose sleep.

I'm interested in using various C/C++ libraries such as ITK to perform image filtering and registration. A significant incompaibiltiy is that ITK uses data pipelines rather than committing pixeldata to memory. My own IDL code actually implements a poor mans version of data pipelines, with a mid-sized granularity (currently frame by frame because it works well with IDLgrWindow::QueryRequiredTiles). After writing a very butchered implementation of pipelines in IDL, I can see this project spinning out of control when I try to write C wrappers for the ITK data model. I also have to realize that ITK isn't the only imaging library which I might want to use.

Shared memory allows me to be a bit more generic and it makes design and testing much easier. The downside is that the space used by all opened images cannot exceed the size of RAM (At least with POSIX anyway). This is problematic if the UI keeps a track of intermediate images.

Robbie