<u>Windows 8 memory management</u>

1. Prerequisite to understand memory management

- Metro style application:-Application buit using Microsoft's new api.
- <u>Memory in windows(you can see in your task manager)</u>

- **Free –** This one is quite simple. This memory has nothing at all in it. It's not being used and it contains nothing but 0s.
- **Available –** This numbers includes all physical memory which is immediately available for use by applications. It wholly includes the Free number, but also includes *most* of the Cached number. Specifically, it includes pages on what is called the "standby list." These are pages holding cached data which can be discarded, allowing the page to be zeroed and given to an application to use.
- **Cached –** Here things get a little more confusing. This number does **not** include the Free portion of memory. And yet in the screenshot above you can see that it is larger than the Available area of memory. That's because Cached includes cache pages on both the "standby list" and what is called the "modified list." Cache pages on the modified list have been altered in memory. No process has specifically asked for this data to be in memory, it is merely there as a consequence of caching. Therefore it can be written to disk at any time (not to the page file, but to its original file location) and reused. However, since this involves I/O, it is not considered to be "Available" memory.
- **Total –** This is the total amount of physical memory available to Windows.

2. Memory management in windows 8

In Windows 8, there are now two Swapfiles: pagefile.sys and swapfile.sys., which makes swap space.

[1]pagefile.sys

Until today, this file was at least as big as your physical memory. Per default, windows sized this file like RAM*1.5. This also was to reserve enough space for the per default enabled kernel memory dump (system properties -> advanced -> startup and recovery).

[2]swapfile.sys

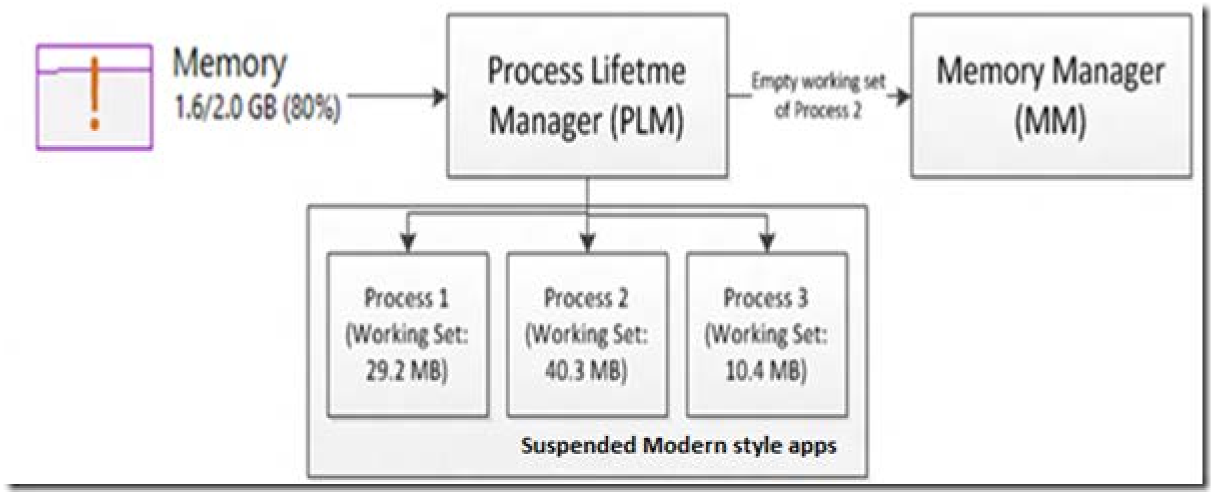
swapfile.sys is a system controller file, normally around 256MB. It's used by Metro style applications that don't fit the traditional paging characteristics (such as usage pattern, growth, space reservation) of pagefile.sys. One example of swapfile.sys usage is the suspend/resume of Metro-style applications;

Swapfile.sys and it's interaction with the Memory Manager

Windows 8 can efficiently write the whole (private) working set of a suspended Modern app to disk in order to gain additional memory when the system detects pressure. This process is analogous to hibernating a specific app, and then resuming it when the user switches back to the app. In this case, Windows 8 takes advantage of the suspend/resume mechanism of Modern apps to empty or re-populate an app's working set.

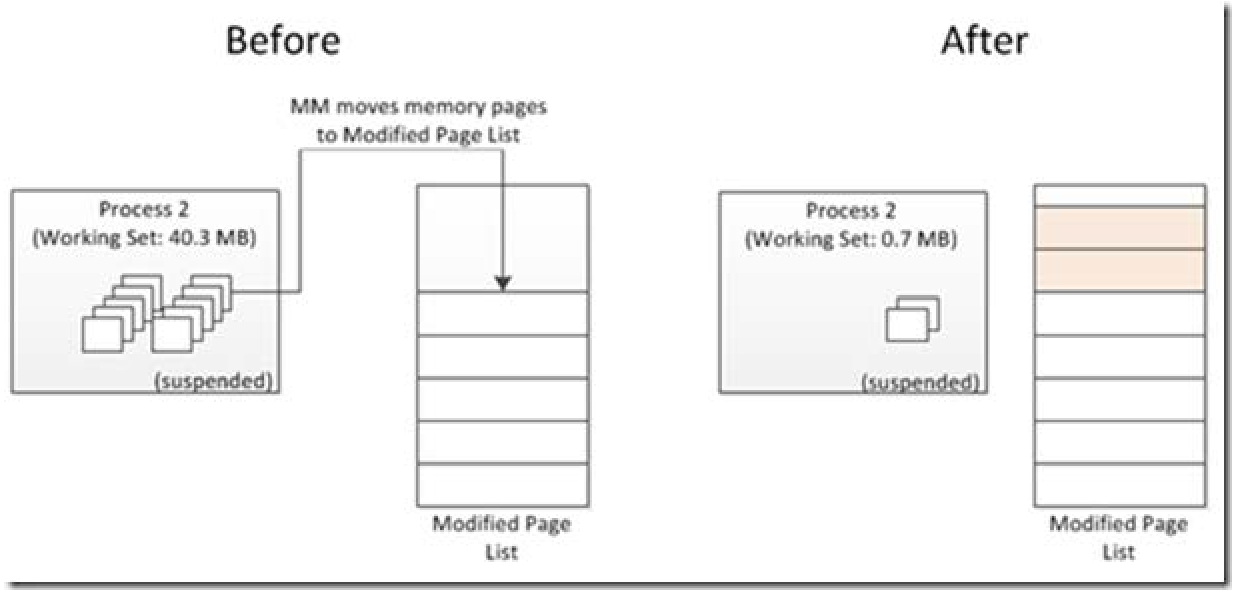
The sequence of events is described below:

1. The Process Lifetime Manager (PLM) detects memory pressure in the system and asks the Memory Manager (MM) to empty the working set of a specific process that houses a suspended Modern style app.



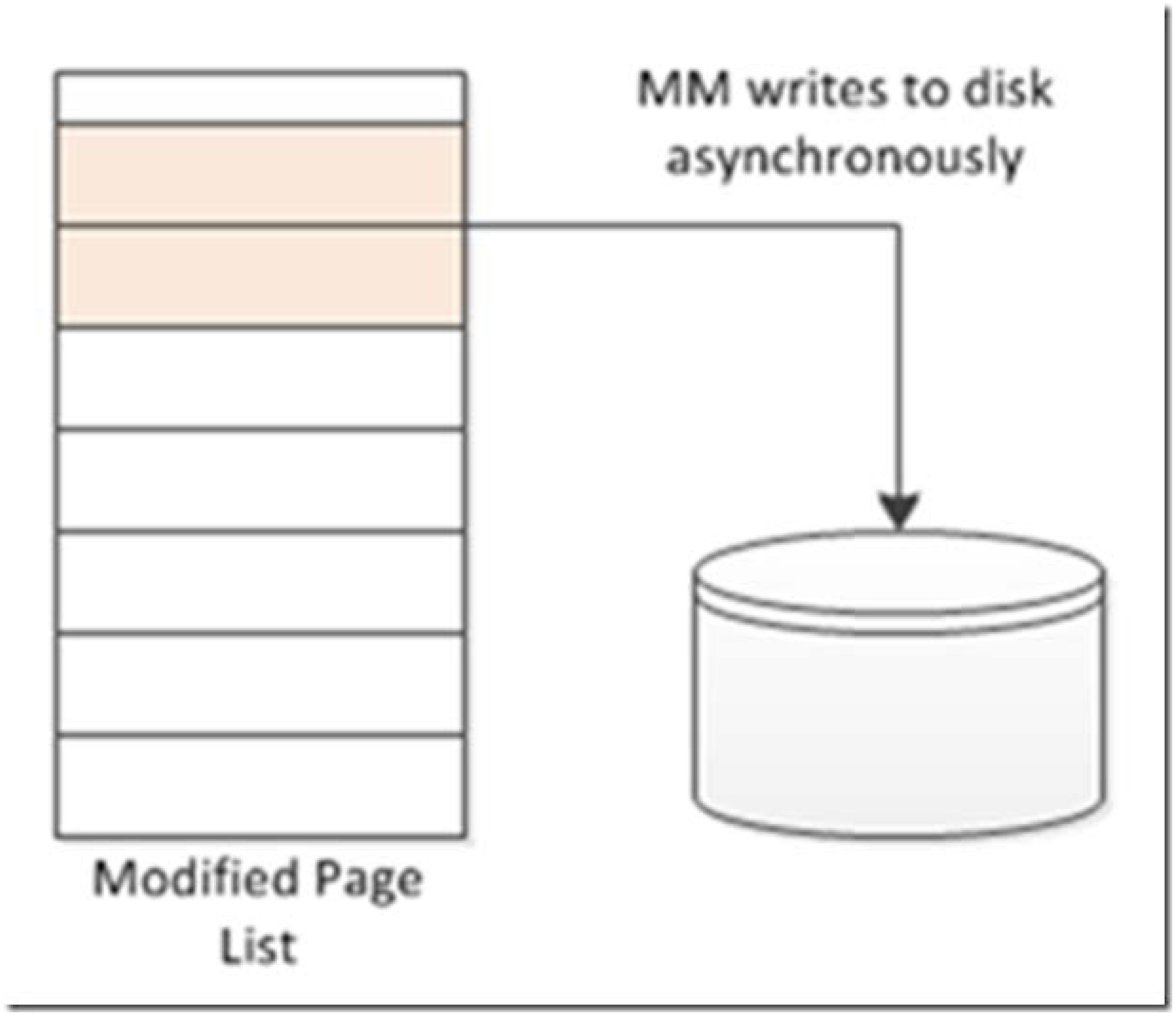
Memory
1.6/2.0 GB (80%)
Process Lifetme Manager (PLM)
Empty working set of Process 2
Memory Manager (MM)
Process 1 (Working Set: 29.2 MB)
Process 2 (Working Set: 40.3 MB)
Process 3 (Working Set: 10.4 MB)
Suspended Modern style apps

2. MM moves the pages of memory from the working set of the app to the operating system's modified page list (which is a list of memory whose contents are to be written out to disk before being reused).

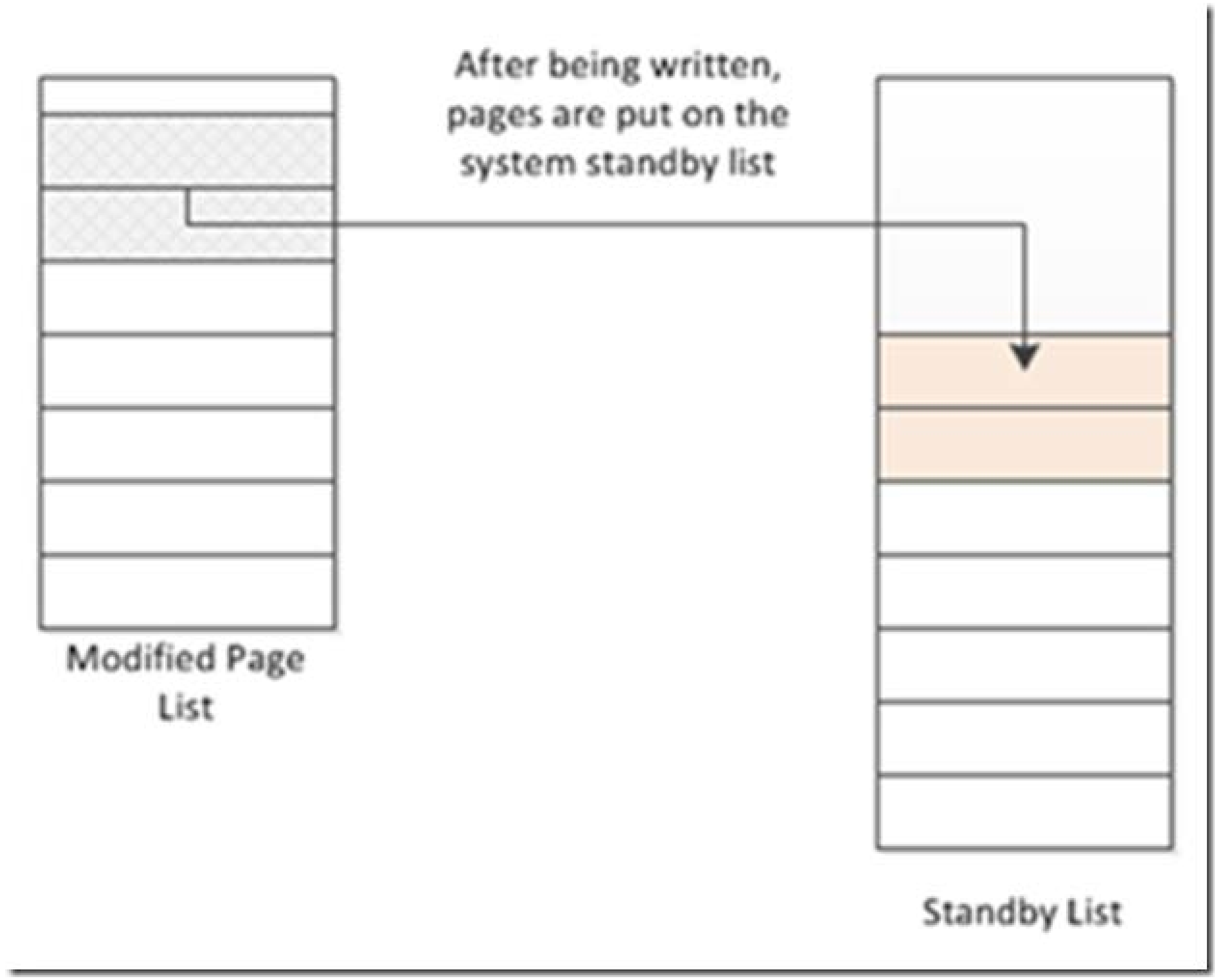


Before
After
MM moves memory pages to Modified Page List
Process 2 (Working Set: 40.3 MB)
(suspended)
Modified Page List
Process 2 (Working Set: 0.7 MB)
(suspended)
Modified Page List

3. The working set pages on the modified page list are written out asynchronously, as dictated by the usual MM policies (written out opportunistically in the background, writes triggered when under memory pressure).



4. Even after the suspended app's working set is written to disk, the memory pages removed from a process are left intact on the operating system's standby list. This is a cache of useful pages of memory that can be repurposed for other apps, if necessary. If these pages are immediately needed again by their original process, they are quickly moved back.


After being written, pages are put on the system standby list
Modified Page List
Standby List

If a user switches back to the app while its working set pages are still in physical memory (on the modified page list or the standby list), the pages will be added back into the app's process immediately. If they are no longer available, Windows will read in the app's working set from disk in an optimized manner.

[3]About hiberfil.sys

This file was as huge as the total amount of RAM installed in your computer. Now it will be sized for 75% of the physical memory but only has 10-15% of that size of data in it. That's because of two main things: (1) the traditional desktop only gets loaded after a user explicitly needs it. This reduces the Memory usage of the Kernel. And (2) Windows Quick-Boot ("clean slate") closes User sessions before shutting the computer down.

To change the size of hiberfil.sys use this command:

```
powercfg /hibernate /size
```

To "full"-shutdown the computer:

```
shutdown /s /full /t 0
```

Windows depends on a considerable number of system services to provide key functionality, and many of this system services start when the system boots and continue to run for as long as the operating system is running. This has two effects; it makes booting take longer, because the services all have to start before the operating system will log in, and these services use system memory.

Start on demand

To counteract this first issue, Windows Vista introduced the notion of a delay-started service. These services still start when the operating system itself is started, but later on in the boot process; they no longer delay the ability to log in and start using the computer

Windows 8 addresses that by including a new "start on demand" model. Services such as Windows Update and Plug-and-Play will only be started when needed—for example, when it's time for the daily check for updates, or when new hardware has been connected—and will stop running when no longer necessary. The result is that, most of the time, memory usage is reduced.

For desktop machines, this is unlikely to provide any real difference in memory usage. However, for those tablets, Microsoft is claiming a net saving of about 23 MB, with the implication that the Metro environment is quite a bit more lightweight than the desktop.

Virtual memory

Windows 8's virtual memory facilities are more powerful than those of previous versions, and the operating system has been altered to make better use of virtual memory.

Virtual memory is the system by which the processor and operating system conspire to lie to applications about the memory in the system. On 32-bit operating systems, the main lie is that every individual process has 2 GiB of private memory available to it, and that memory is linear: every byte of that memory has an address and these addresses are contiguous, starting at zero, and going all the way up to 2^{31}. This address space is larger on 64-bit systems, but the basic principles are the same.

The processor divides this address space into blocks called pages; usually each page is 4 kiB, though other sizes are possible in certain circumstances. Pages are the units that the operating system mainly deals with; whenever memory is "paged out"—that is, written to disk, to free up physical memory—this activity happens a page at a time. Conversely, when data needs to be "paged in"—read from disk into physical memory—that too happens with page granularity.

Almost all memory in Windows can be paged out to disk. it's where most pages are placed when they're not resident in physical memory. However, not everything gets written to the pagefile. Most operating systems, including Windows, have a concept of memory-mapped files. Memory-mapped files allow the creation of pages of memory that correspond to specific named files in the filesystem. When these pages are paged out, they don't get written to the pagefile; they get written to the specific mapped file. Better than that, the pages only get written if they have been modified. If they have not been altered since they were read from the file, Windows doesn't have to write the pages back out; it can just discard them. If it ever needs the pages again, they can be safely reread from the file.

Memory-mapped files are most commonly used for loading program executables and DLLs; Windows creates memory-mapped files for each EXE and DLL. These mappings are almost always read-only. Page in operations hence come from the EXE and DLL files, and for page out operations, Windows can simply discard the pages.

Sharing pages

Memory pages can be shared between different processes.

Again, memory-mapped files are the most common candidates for this kind of sharing; if two different processes both have the same DLL memory mapped, the pages of that DLL don't need to be duplicated. Since their content is the same—because it originates from the same DLL—the same physical memory can be used. The result is that while system DLLs are loaded by almost every process on the system, they only need to be loaded into memory once.

This memory sharing is useful, especially on low-memory systems, but it has its limits. The big one is that Windows only shares memory that corresponds to memory-mapped files. That's because this is the only time that Windows *knows* that the pages are all *identical*.

For regular data—anything that gets written to the pagefile—there's no sharing.

Memory Combining

Windows 8 will include a new mechanism to allow these pages to be shared. The system will periodically scan memory, and when it finds two pages that are identical, it will share them, reducing the memory usage. If a process then tries to modify a shared page, it will be given its own private copy, ending the sharing.

Splitting data structures

Another change being made is a reorganization of internal data structures used by the operating system. Data that isn't used often is a good candidate for being paged out, in order to make more physical memory available. However, if even a single byte of a page stored in the pagefile is needed, the entire page has to be read from disk. If the data the operating system needs is scattered throughout memory—a few bytes in one page, a few more in the next, and so on—Windows is unable to move those pages to disk, thereby increasing the memory load.

In Windows 8, Microsoft says it has tried to split data structures so that data that is used often is kept separately from data that's used only infrequently; this way, the parts that aren't used often can be readily moved out to disk. This work involved modifying low-level components that are some of the very oldest parts of the operating system. Surprisingly, the company says that it did this work two years ago, and has been testing it on employee desktops since then. It was done so early in Windows 8's development to ensure that there was plenty of time for testing and gathering of data.

The results Redmond is claiming are impressive; consolidating often-used data on average saves "tens of MB" per machine. That's a big gain for what's essentially just a neater, tighter packing of data.

When an operating system does run out of physical memory, and has to page information out to disk, it has something of a problem: it doesn't know which memory is the best thing to page out. The best page to move to disk is the one that is least likely to be accessed (that is, the one that will be able to stay on disk as long as possible), but the operating system cannot predict the future, so it can never know for sure which page it should pick. Typically, operating systems guess: they assume that the pages that have not been used recently will probably not be used in the future either. These **least-recently used pages** are then preferentially written out to disk to make more physical memory available.

Least-recently used is a good guess, but it's not always right. Windows 8 will provide a new way for applications to give hints to the virtual memory system to mark some memory as "low priority." Low priority memory will be preferentially paged out *even if it has been recently used*.

The example the company gives is of an anti-virus application. The malware scanner might need to allocate memory during a scan, but scans are relatively infrequent. This makes the scanner's memory a good candidate for being "low priority"; the system can write it out to disk aggressively, because it's *probably* not going to be needed, even if the memory has been recently used.

Taken together—services and a desktop that don't run unless needed, memory deduplication, memory prioritization, and better packing of operating system structures—Microsoft is claiming that Windows 8 could end up with memory usage 100 MB or more lower than Windows 7's, on the same hardware. For desktops and laptops, this may not be such a big deal, but of course, Microsoft has its eye on tablets. Every improvement made in Windows 8 will apply to ARM as well as x86 hardware, and reducing the memory load on ARM machines is critical. The work done on Windows 8 should make running on 512 MB machines a practical possibility; still tight, but viable.

Reference

[1]web reference o n http://arstechnica.com/information-technology/2011/10/how-windows-8s-memory-management-modifications-make-for-a-better-user-experience/

[2] http://768kb.wordpress.com/2012/11/07/windows-8-memory-management/

[3] http://blogs.msdn.com/b/b8/archive/2011/10/07/reducing-runtime-memory-in-windows-8.aspx

[4] http://www.askvg.com/comparison-between-windows-7-and-windows-8-memory-management-system/

[5] http://www.neowin.net/news/microsoft-details-windows-8-memory-usage-reduction

[6] http://blogs.msdn.com/b/tims/archive/2010/10/29/pdc10-mysteries-of-windows-memory-management-revealed-part-two.aspx