Johannes Joujo

Frågeställning

Hur mycket minne kan en process maximalt använda?

Hur och varf¨or skiljer sig v¨ardena fr˚an de olika verktygen? Vilka teoretiska och praktiska gr¨anser finns f¨or minnesallokering? H¨anvisa till externa k¨allor. Hur f¨or¨andrades minneskompositionen under k¨orning och varf¨or? Vad har virtuellt minne f¨or effekt?

Analys

En process kan ha tillgång till olika mycket minne beroende på målplattform, om LAA är påslaget och hur mycket minne som datorn redan använder för andra processer. Enligt resultatet kan en 32-bitars målplattform utan LAA allokera 2 GB till en process, och 4 GB till en process när LAA är påslaget, det stämmer med vad [1] säger att en 32-bitars målplattform ska i teorin kunna allokera till en process. Enligt [1] är den maximala för en 32-bitars målplattform är 4 GB med LAA och 2 GiB utan. Resultatet visar att när det allokeras 1 KiB till 32-bitars målplattform allokeras 1,82 GiB, vilket överensstämmer med det teoretiska.

För en 64 målplattform utan LAA är maximala 2 GB, när minnets storlek allokeras I KiB står det i terminalen att 1,81 GB har allokerats och 1 GB när den allokeras med 1 GiB. Resultatskillnaden bero på att den andra GB eller KiB som systemet vill allokera till processen är för stor på grund av att det finns andra grejer i registorn än minnesadressen vilket leder till att det som visas i terminalen är hur mycket minne som har allokerats till procesen, det som visas i visual studio process memory är den totala för processen (det som allokerats + default värde).

För en 64 målplattform med LAA påslaget och allokerar 1 KiB är maximala 34,58 GB och sen kraschar hela datorn. Med LAA är allokeras 37 GB. Enligt [2] kan 64-bitars målplattform allokera teoretiskt 128 TB. Programmet kraschar när 1 KiB allokeras oändligt mycket eftersom den kommer till en punkt då den har allokerat så mycket minne att datorn kraschar. Datorn kraschar inte när det allokeras 1 GiB oändligt mycket eftersom programmet kommer komma till en punkt då det inte går att allokera mer minne, då blir det bad alloc men datorn överlever eftersom det finns fortfarande minne för den att vara funktionell.

Taskmanagers minnes kompositionen är uppdelat i olika delar:

In use- minne som används av processer, drivers, eller andra operating systems, det finns även in use compressed som sparar systemet lite minne.

Modified- minne som har ändrats och måste skrivas till disken innan minnet kan användas för ett annat syfte.

Standby- Minne som innehåller cached data som inte använd just nu.

Free- minne som inte är i in use och kommer att allokeras när processer, drivers eller operating system behöver mer minne.

Det som händer när en av processerna körs är att in use växer tills en punkt då den behöver använda det som är modified och då skrivs informationen till disken innan den kan använda minnet, till slut kommer den komma till en punkt då den behöver hela tiden frigöra från modified och då stoppas programmet.

Virtuellt minne är sekundär minne som kan användas som backup för primärminnet. Det är långsammare att använda sekundär minnet jämfört med primär minnet men ibland är det nödvändigt. När primär minnet är fullt och det behövs mer minne överförs data från primärminnet till sekundär minnet. Det gör att mer minne kan användas till processen som körs. Det går inte att överföra hur mycket data som helst till sekundär minnet, det går att ställa in manuellt hur mycket som ska gå att allokera till sekundär minnet men i skolans datorer är de begränsade till 16 GB. [3]

Tabell 1 visar alla resultat.

Målplatt form	LAA on/off	Terminal resultat (GB)	In use (compressed) (GB)	Chuck storlek	Error message	VS process memory (GB)	Committed (GB)	Time (s)
32	off	1,82	7,8	2^10	Bad_allocation	2	9,7	3,3
32	on	2	7,8	2^30	Bad_allocation	2	9,8	0,8
64	off	1,81	8	2^10	Bad_allocation	2	9,7	3,1
64	on	37	14,9	2^30	Bad_allocation	38,1	44,9	31,4
32	off	1	6,8	2^30	Bad_allocation	1	8,6	0,4
32	on	3,65	9,6	2^10	Bad_allocation	4	11,7	8
64	off	1	7.5	2^30	Bad_allocation	1	9.0	0,5
64	on	34,58	15,5	2^10	Stängs av	34,3	45,0	106,2

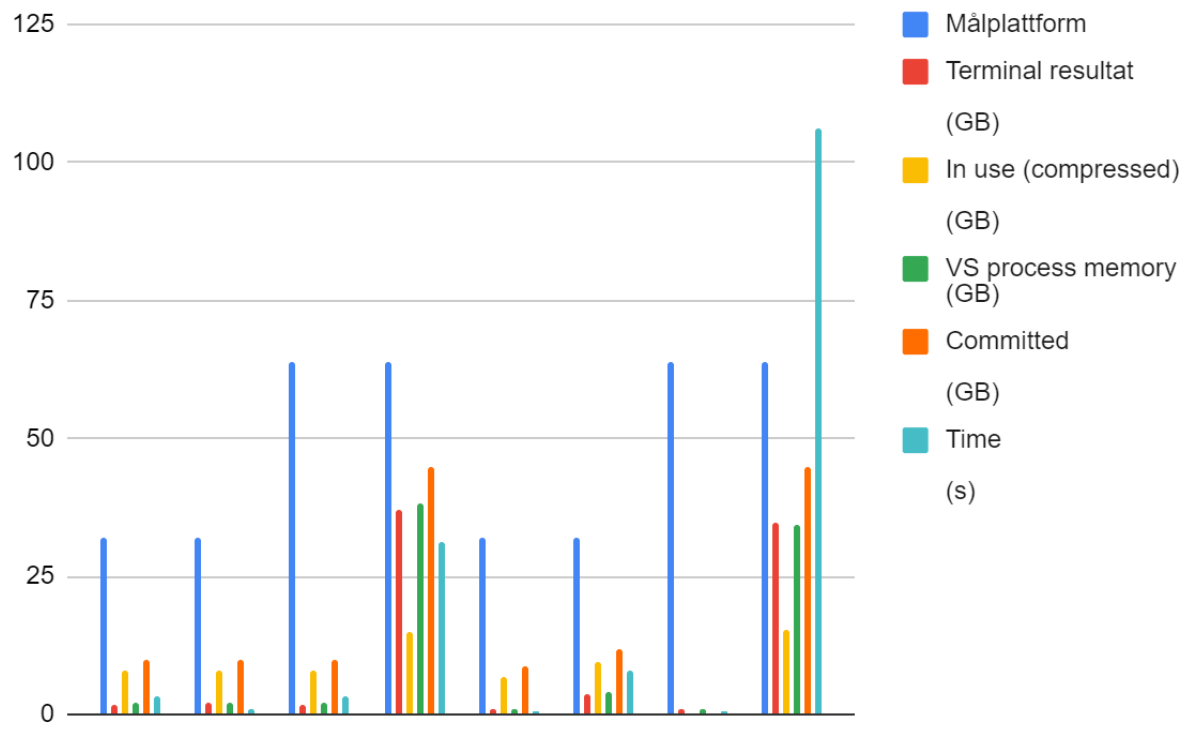

Diagram 1 visar stapel diagram med värden från tabell 1.

Tabell 2 visa resultatet för 32 och 64 bitars målplattform när LAA är avstängt.

Målplatt form	LAA on/off	Terminal resultat (GB)	In use (compressed) (GB)	Chuck storlek	Error message	VS process memory (GB)	Committed (GB)	Time (s)
32	off	1,82	7,8	2^10	Bad_allocation	2	9,7	3,3
64	off	1,81	8	2^10	Bad_allocation	2	9,7	3,1
32	off	1	6,8	2^30	Bad_allocation	1	8,6	0,4
64	off	1	7.5	2^30	Bad_allocation	1	9.0	0,5

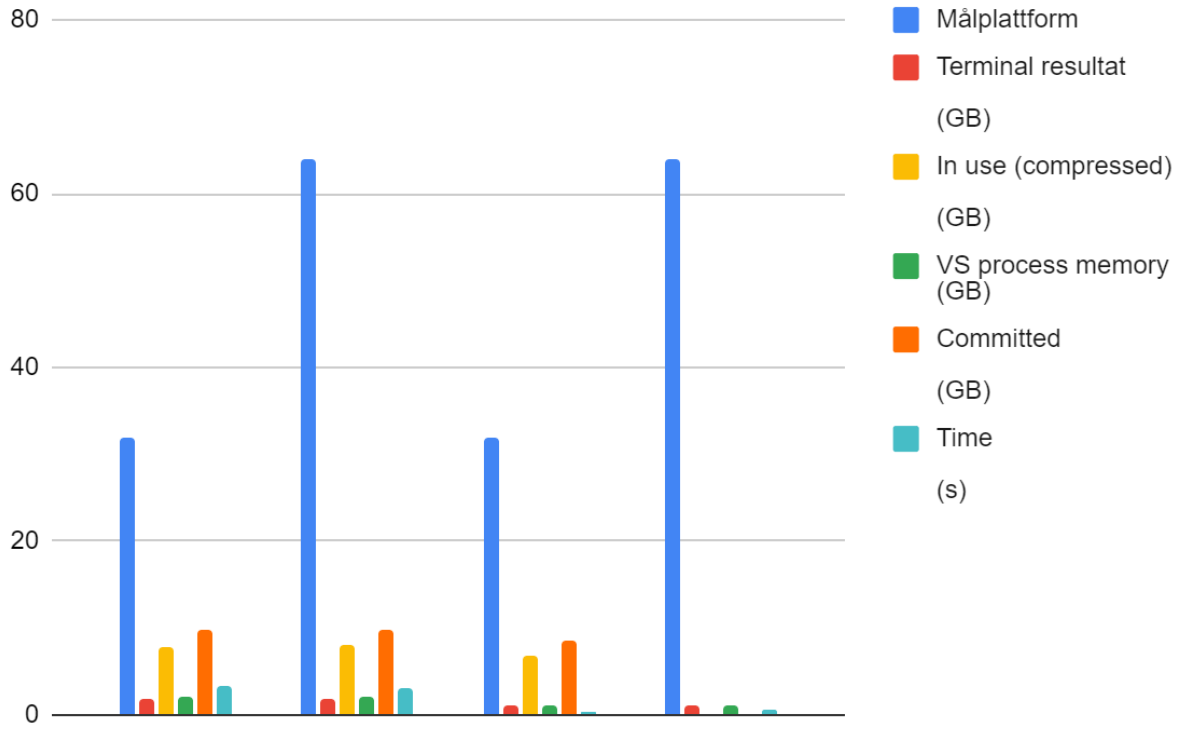

Diagram 2 visar stapeldiagram för värden från tabell 2.

Tabell 3 visar resultatet för 32 och 64 bitars målplattform när LAA är påslaget.

Målplattform	LAA on/off	Terminal resultat (GB)	In use (compressed) (GB)	Chuck storlek	Error message	VS process memory (GB)	Committed (GB)	Time (s)
32	on	2	7,8	2^30	Bad_allocation	2	9,8	0,8
64	on	37	14,9	2^30	Bad_allocation	38,1	44,9	31,4
32	on	3,65	9,6	2^10	Bad_allocation	4	11,7	8
64	on	34,58	15,5	2^10	Stängs av	34,3	45,0	106,2

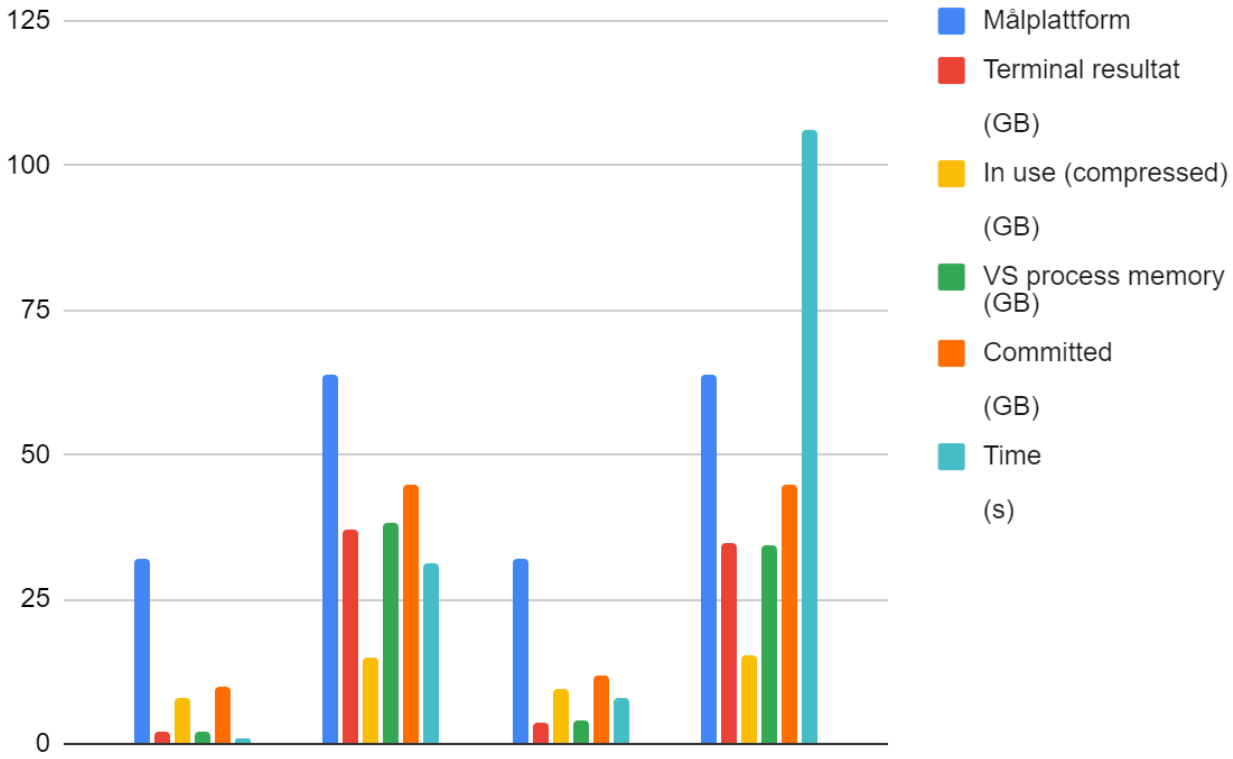

Diagram 3 visar stapeldiagram för värden från tabell 3.

Källor

[1]" Large Address Aware capability change for Excel", information tagen 1 dec 2022. [Online]. Available: https://learn.microsoft.com/en-us/office/troubleshoot/excel/laa-capability-change

[2] "Memory Limits for Windows and Windows Server Releases", information tagen 1 dec 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/memory/memory-limits-for-windows-releases

[3]” Hur du ökar det virtuella minnet på din Windows 10” 11 okt 2019 https://itigic.com/sv/increase-virtual-memory-windows-10/