IBM Enterprise2013
pST587 - Optimizing IBM Tivoli Storage Manager Performance on AIX

Grover Davidson – grover@us.ibm.com

Enterprise2013

Topics

- Tivoli Storage Manager (TSM)
- IBM ATape Device Driver
- Virtual Memory Manager (VMM)
- Memory Pages and Pinning
- Problem Determination
- Making Changes

Optimal TSM Performance

- TSM prior to version 6.1 uses a proprietary database.
- Starting with release 6.1, TSM uses a DB2 database.
- Essential configuration and tuning is required for:
  - AIX file system
  - Disk subsystems for TSM database, recovery logs and storage pools
  - TSM domain policies (proper data layout)
  - DB2
- These are outside the scope of this document.

VMM and ATape

- Atape is the IBM Advanced Tape Driver.
- ATape reads/writes directly to hardware and virtual adapters.
- Because these buffers will be used with an adapter, the memory must be pinned. This applies to both physical and virtual adapters.
- Pinning is done by acquiring the SCB (VMM Segment Control Block) lock and then pinning 1 page. The lock is released after each page is pinned and acquired again.
- A typical TSM buffer size for a tape drive is 256K and the memory comes from TSM data space. The buffer pinned/unpinned on each IO call.
  - Pinning 256K of memory on 4k pages requires 64 separate cycles of acquiring the SCB lock, pin the page, and release the SCB lock.
  - If 64K pages are used, 4 cycles are required.
  - If 16M pages are used, only 1 cycle is needed.

Page size support

Page size	Pages to address 256k of memory	Processor requirements
4K	64	Supported on all platforms
64K	4	Supported on Power5+/Power6/ Power7
16M	1	Supported on Power4 or higher

VMM Segment Locking and Memory Pinning

- When 2 processes are trying to pin pages, they are normally using 2 different segments for the process data segment and as a result for 2 processes there is no issue with VMM segment locking.
- Shared memory is the exception since the same segment can be used by more than 1 process.
- When several threads in a single process are pinning pages for buffers, the buffers are usually in the same memory segment.
- As a result, lock contention only occurs when multiple threads in a single process are pinning data.
- TSM processes are multithreaded and use one thread per tape device.

Pinning Memory

Pseudo code for pinning memory:

```
while (pages to pin)
do
        lock VMM segment
        pin page
        unlock VMM segment
done
```

The amount of memory pinned on each pass is defined by the page size.

The lock/unlock on a per page basis is done to allow fair access for other threads to the VMM segment lock.

For a single thread, there is not real lock contention.

The time to acquire and release the VMM lock is not significant without lock contention.

2 Threads pinning memory

With 2 threads in the same process, we now have contention for the VMM lock.

Each thread will take turns waiting to acquire the lock while the other thread pins a page. As more threads perform this, the wait time gets longer.

2 threads pinning memory shows us the lock contention:

thread 1	thread 2
lock VMM segment	
pin page	**wait on VMM segment lock**
unlock VMM segment	
	Lock VMM segment
wait on VMM segment lock	Pin page
	Unlock VMM segment
lock VMM segment	
	wait on VMM segment lock

After the IO is completed by ATape the buffers also need to be unpinned using the same logic.

Data Collection

There are several types of data used for this analysis. The easiest way to collect what is needed is to run perfpmr:

perfpmr.sh 600

vmstat data is located in the monitor.int file.

svmon data is in 'svmon.before' and 'svmon.after' files.

splat data takes 2 commands to generate. First we merge sort the lock trace files with:

trcrpt –r –C all trace.raw.lock > trace.lock.tr

And then generate splat output:

splat –i trace.lock.tr –n trace.syms –da –p –o splat.out

If your system is NOT SMT capable, you should not use the '-p' flag above.

The formatted lock trace file is generated with:

*trcrpt –x –t trace.fmt –n trace.syms –O tid=on,cpu=on,PURR=on –o trace.lock.int *
trace.lock.tr

perfpmr can be obtained from ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr.

Please download the correct version for your level of AIX EACH TIME YOU USE IT!

Perfpmr tool is updated frequently (sometimes daily) as changes are made to it.

How 4k pages look in vmstat

- vmstat will show a high %sys time even when the system is relatively idle:

kthr			memory		page		faults			cpu						
r	b	p	avm	fre	fi	fo	in	sy	cs	us	sy	id	wa	pc	ec
21	2	0	1638632	2317759	15364	9	3940	44120	26796	4	32	62	1	2.22	37.0
16	2	1	1638622	2317928	16079	18	4048	38769	27861	4	33	62	1	2.24	37.3
22	2	0	1638766	2317458	13969	10	3629	36405	25692	4	32	64	1	2.16	36.0
23	2	1	1638783	2317416	15152	7	4419	42391	30054	4	34	61	1	2.31	38.5
20	2	0	1638881	2317222	14857	9	4074	45549	27943	4	33	62	1	2.25	37.5
22	2	0	1638888	2316928	13062	9	4030	37084	26850	4	33	63	1	2.23	37.2
21	1	1	1638878	2317026	16012	7	4698	44032	31096	4	34	60	2	2.33	38.9
21	1	1	1638888	2316994	14750	19	4222	46657	28350	4	33	62	1	2.28	37.9
19	2	0	1638893	2316895	14218	4	3809	37162	26646	4	33	62	1	2.23	37.1
18	2	1	1638914	2316519	16080	9	4228	41577	29581	4	33	61	2	2.29	38.1

- Note that %sys is several times that of %usr. This indicates we are not doing much work on behalf of the application.
- These are 10 second intervals.
- Vmstat data is near the bottom of the monitor.int file collected by perfpmr.

What tprof shows

- If tprof data is collected and reviewed, we see lots of system time in the application (user + shared) and the kernel time shows us the time is in locking routines:

```
Process                                Freq  Total Kernel   User Shared  Other   Java
=======                                ====  ===== ======   ==== ======  =====   ====
/usr/bin/dsmserv                        137  81.31  74.91   5.45   0.95   0.00   0.00
wait                                     24  10.48  10.48   0.00   0.00   0.00   0.00
/usr/bin/dsmc                             6   7.35   7.21   0.09   0.05   0.00   0.00
/usr/java14/jre/bin/java                 25   0.20   0.12   0.00   0.06   0.03   0.00
.
.
.
11  Total % For All Processes (KERNEL) = 91.03

Subroutine                                                %   Source
==========                                           ====== ======
.unlock_enable_mem                                    59.88 64/low.s
h_cede_end_point                                       8.04 hcalls.s
h_put_tce_end_point                                    4.36 hcalls.s
pcs_glue                                               2.94 vmvcs.s
.enable                                                2.31 misc.s
```

- The kernel time for dsmserv is 74.91% of the total 81.31% CPU time used.
- The ‘unlock…’ routine in the kernel shows that time is being spent in lock related code.

SPLAT (SPin Lock Analysis Tool) results

- No one lock will stand out as an issue:

Lock Name, Class, or Address	Type	Acquisitions or Passes	Spins	Wait or Transform	%Miss	%Total	Locks or Passes / CSec	Percent Holdtime Real CPU	Real Elapse	Comb Spin
F10007165096B6F8	D	28925	3314	0	10.2795	14.5338	69203.267	4.9881	8.4728	0.6213
F1000716509DDF08	D	59	10	0	14.4928	0.0296	141.158	4.2299	8.1282	0.3849
F1000716509DE388	D	75	17	1	18.4783	0.0377	179.438	5.8068	10.7485	0.3729
F1000716509E1A18	D	83	14	0	14.4330	0.0417	198.578	5.0160	9.3680	0.3403
F100071650963FB8	D	5330	1278	0	19.3402	2.6781	12752.063	0.9628	1.6130	0.2162
F1000716509E00C8	D	29	7	1	19.4444	0.0146	69.383	1.3904	2.8325	0.1054
F100060046ED8600	D	1727	60	0	3.3576	0.8678	4131.860	0.9865	1.6686	0.0571
F100060046ED8780	D	1985	52	0	2.5528	0.9974	4749.127	1.0927	1.8385	0.0554
F100060046ED8140	D	1525	49	0	3.1131	0.7663	3648.573	0.8073	1.3881	0.0425
F100060046ED8180	D	2381	78	0	3.1720	1.1964	5696.559	0.7070	1.1961	0.0293

No single lock has a high Combined Spin time. Spin time is a measurement of how long AIX spins continuously trying to acquire the lock.

SPLAT (Simple Performance Lock Analysis Tool) results

- Checking the details of one of the locks listed will show locking occurred with interrupts disabled:

Lock Activity (mSecs) - **Interrupts Disabled**

SIMPLE	Count	Minimum	Maximum	Average	Total
LOCK	28925	0.000125	0.004750	0.000721	20.848719
w/ KRLOCK	0	0.000000	0.000000	0.000000	0.000000
SPIN	3314	0.000125	0.007781	0.000783	2.596844
KRLOCK LOCK	0	0.000000	0.000000	0.000000	0.000000
KRLOCK SPIN	0	0.000000	0.000000	0.000000	0.000000
TRANSFORM	0	0.000000	0.000000	0.000000	0.000000

Function Name	Acqui-sitions	Miss Rate	Spin Count	Transf. Count	Busy Count	Percent Held of Total Time CPU	Elapse	Spin	Transf.
.pin_seg_range	28480	10.33	3282	0	0	4.95	8.41	0.62	0.00
.as_geth	445	6.71	32	0	0	0.04	0.07	0.00	0.00

ThreadID	Acqui-sitions	Miss Rate	Spin Count	Transf. Count	Busy Count	Percent Held of Total Time CPU	Elapse	Spin	Transf.	ProcessID	Process Name
7495773	6045	13.88	974	0	0	16.89	1.73	2.99	0.00	1011752	dsmserv
7053429	1690	15.12	301	0	0	16.18	0.55	2.68	0.00	1011752	dsmserv
7069823	325	9.47	34	0	0	19.23	0.11	2.57	0.00	1011752	dsmserv
7389409	6045	12.42	857	0	0	17.08	1.75	2.48	0.00	1011752	dsmserv
6860935	6045	9.98	670	0	0	17.42	1.75	2.42	0.00	1011752	dsmserv
7360701	6045	6.05	389	0	0	17.85	1.74	1.21	0.00	1011752	dsmserv
6942813	1430	3.96	59	0	0	19.04	0.44	0.84	0.00	1011752	dsmserv
7090311	1300	2.26	30	0	0	18.40	0.40	0.41	0.00	1011752	dsmserv

- And show the function acquiring the lock as a pin routine – pin_seg_range in this case.
- Interrupts disabled means the thread will not go to sleep and wait for the lock. In kernel mode threads spin on the lock.

Formatted lock trace

- Formatted lock trace will show this pattern of missed locks:

```
Thread ID     Action          Results
7069823       lock:           dlock lock addr=F10007165096B6F8       Lock is acquired
6860935       lock:           dmiss lock addr=F10007165096B6F8       Missed the lock and have to wait
7069823       unlock:    lock addr=F10007165096B6F8                  Lock is released
6860935       lock:           dlock lock addr=F10007165096B6F8       Waiting thread gets lock
7069823       lock:           dmiss lock addr=F10007165096B6F8       Fist thread now misses the lock and waits
6860935       unlock:    lock addr=F10007165096B6F8 lock
7069823       lock:           dlock lock addr=F10007165096B6F8
6860935       lock:           dmiss lock addr=F10007165096B6F8
7069823       unlock:    lock addr=F10007165096B6F8 lock
6860935       lock:           dlock lock addr=F10007165096B6F8
7069823       lock:           dmiss lock addr=F10007165096B6F8
6860935       unlock:    lock addr=F10007165096B6F8 lock
7069823       lock:           dlock lock addr=F10007165096B6F8
6860935       lock:           dmiss lock addr=F10007165096B6F8
7069823       unlock:    lock addr=F10007165096B6F8 lock
6860935       lock:           dlock lock addr=F10007165096B6F8
7069823       lock:           dmiss lock addr=F10007165096B6F8
6860935       unlock:    lock addr=F10007165096B6F8 lock
7069823       lock:           dlock lock addr=F10007165096B6F8
6860935       lock:           dmiss lock addr=F10007165096B6F8
7069823       unlock:    lock addr=F10007165096B6F8 lock
6860935       lock:           dlock lock addr=F10007165096B6F8
7069823       lock:           dmiss lock addr=F10007165096B6F8
6860935       unlock:    lock addr=F10007165096B6F8 lock
7069823       lock:           dlock lock addr=F10007165096B6F8
6860935       lock:           dmiss lock addr=F10007165096B6F8
```

Segment information from svmon

```
Svmon -P 999626

-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
  999626 dsmserv         264607    66591        0   261710      Y     Y     N

     PageSize       Inuse        Pin       Pgsp     Virtual
     s   4 KB      196863       1007          0      193966
     m  64 KB         138          3          0         138

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
  138925        12 work text data BSS heap           s  65536     0    0 65536
       0         0 work kernel segment (lgpg_vsid=0) L     16    16    0    16
  34086a        11 work text data BSS heap           s  62072     0    0 62072
  198931        13 work text data BSS heap           s  49695     1    0 49695
  2e0a5e        14 work text data BSS heap           s  10355     0    0 10355
   38905         - work                              s   4005  1006    0  4005
  3a8777        10 clnt text data BSS heap,          s   2355     0    -     -
                        /dev/tsmwin_lv:4106
  1870b0  90000000 work shared library text          m    133     0    0   133
    8001  9ffffffd work shared library               s   1786     0    0  1786
  2e08de  9001000a work shared library data          s    192     0    0   192
  258249  90020014 work shared library               s    158     0    0   158
  1a08b6 f00000002 work process private              m      5     3    0     5
   78b0d  80000000 work private load text            s     51     0    0    51
  3208e6  8fffffff work private load data            s     33     0    0    33
  2f88dd  80020014 work USLA heap                    s     27     0    0    27
   68b0f  8001000a work private load data            s     22     0    0    22
   e851f  ffffffff work application stack            s     18     0    0    18
  2300c6  9ffffffe work shared library               s     16     0    0    16
  150028  9fffffff clnt USLA text,/dev/hd2:2392      s     14     0    -     -
```

Note that most segments are using small pages (PSize=s). These are 4k pages. 'm' indicates 64k pages and 'L' is for 16MB pages. We do not want segments with multiple page sizes if possible.

SELECTING A PAGE SIZE

If you have a Power4 system, you can only use 16MB pages. 64KB pages are not available on this hardware.

64KB pages are managed by AIX, require actual pin/unpin operations, can be pushed out to paging space in an emergency.

16MB pages are explicitly created, pinned by design, and limited in the quantity. If TSM needs a 16MB page and they are marked a 'mandatory', TSM will fail with a memory error.

16MB pages can be marked as 'use if available but allow other pages if no 16MB pages are available'. This prevents TSM from failing due to a lack of 16MB pages.

64KB pages provide the best performance and maximum flexibility.

16MB pages provide maximum performance but at the price of flexability.

LDR_CNTRL environment variable

- Environment variable to control page sizes used by an application.
- Completely transparent to the application unless 16MB pages are depleted and mandatory.
- 3 Main options to consider:
  - 64K pages only
  - 16MB pages only
  - 16MB pages if available and 64KB pages otherwise
- Because we are optimizing the page sizes, we will also optimize the text, stack and data pages sizes to use 64KB pages.
  - TEXTPSIZE=64K
    - Use 64KB pages for code.
  - STACKPSIZE=64K
    - Use 64KB pages for the stack.
  - DATAPSIZE=64K
    - Use 64KB pages for process heap/data.
  - SHMPSIZE=64K
    - If shared memory is used, use 64KB pages. If LARGE_PAGE_DATA is set

Setting Large Page Usage

- Value name is LARGE_PAGE_DATA
- This is the KEY tunable to prevent lock contention while pinning/unpinning the pages.
- Value of M means mandatory.
  - The value of M makes large pages for heap and data segments mandatory. If the application allocates memory and there are no large pages left, malloc returns ENOMEM.
- Value of Y means use if available
  - Use large pages for data and heap if possible, but if none are available check other LDR_CNTRL variables for guidance and then fall back to default behavior.
  - TSM will not crash due to lack of 16MB pages.
- Do NOT specify DATAPSIZE if LARGE_PAGE_DATA is specified

Assembling LDR_CNTRL

- The first part is fixed (unless Power4 processor is being used):
  - LDR_CNTRL=TEXTPSIZE=64K@STACKPSIZE=64K@SHMPSIZE=64K
- Next, choose the option for 16MB pages. Y is the recommended values to prevent TSM crashes due to lack of 16MB pages:
  - LARGE_PAGE_DATA=Y
- Put it all together:

  LDR_CNTRL=TEXTPSIZE=64K@STACKPSIZE=64K@SHMPSIZE=64K@LARGE_PAGE_DATA=Y
- OR

  LDR_CNTRL=TEXTPSIZE=64K@STACKPSIZE=64K@SHMPSIZE=64K@DATAPSIZE=64K
- Do not set this in the /etc/environment file! Bad things will happen.
- Do not set this in the general TSM user environment! Bad things will happen.
- The correct place to put this setting is in the TSM startup script.

Determine how many large (16MB) pages we need

1. Let dsmserv or dsmsta process run for a week or longer.
2. Run ‘svmon –P <PID>’ and count the number of work segments with ‘BSS heap’

```
# svmon -P 2687182
     Pid Command                    Inuse      Pin     Pgsp
 2687182 dsmserv                    426493     9136    60847

    Vsid      Esid Type Description               PSize
 1160816        14 work text data BSS heap            s
 10f080f        12 work text data BSS heap            s
 1110811        13 work text data BSS heap            s
 10b04cb        11 work text data BSS heap            s
 10f082f        15 work text data BSS heap            s
 10c03ec        16 work text data BSS heap            s
 10f0fef        17 work text data BSS heap            s
 1100ef0        18 work text data BSS heap            s
```

3. The number of pages to start with is the number of segments * 16 + 32 extra pages. This will allow 16 pages for the kernel and some more for growth.
4. In this example, we have 8 segments * 16 pages/segment + 32 pages = 128 + 32 = 160 pages as our initial guess.

Configuring large pages

- Configure the number of large pages in AIX by running (assumes 0 large pages already configured):

  vmo -p -o lgpg_regions=160 -o lgpg_size=16777216
  bosboot –a
  reboot

  You should over configure and then reduce the configured pages based on actual usage.

- The following messages in the TSM log indicate that you do not have enough large pages configured if 16MB pages are mandatory:

  ANR9999D Memory allocation error.
  ANR0358E Database initialization failed: sufficient memory is not available.

  – Configure more large pages by increasing the lgpg_regions value.

Enabling TSM user to use large pages

If TSM is not running as root, enable the TSM user id to use large pages:

chuser capabilities=CAP_BYPASS_RAC_VMM,\
CAP_PROPAGATE,CAP_NUMA_ATTACH user_id

Where user_id is the user_id TSM is running under.

Without this chuser step, a non-root user will not be able to use large pages.

Failure to do this will prevent TSM from using 16MB pages.

Edit the TSM startup

- Then modify the startup file (/usr/tivoli/tsm/server/bin/rc.adsmserv) and change:

  dsmsta quiet

  To:

  LDR_CNTRL=TEXTPSIZE=64K@STACKPSIZE=64K@SHMPSIZE=64K@LARGE_PAGE_DATA=M dsmsta quiet

  NOTE: It may be dsmsta OR dsmserv depending on this being a TSM storage agent or the TSM server.

- **Restart the server using the startup file!**

Verifying things worked

- Run ‘svmon –P’ on the new TSM server process:

```
 Pid        Command Inuse    Pin     Pgsp    Virtual 64-bit Mthrd 16MB
2978016    dsmsta 89962  76492       0      89960      N      Y     M

PageSize Inuse  Pin  Pgsp  Virtual
s 4 KB       42  12     0       40
m 64 KB    1012 172     0     1012
L 16 MB      18  18     0       18

Vsid      Esid Type Description                      PSize Inuse  Pin Pgsp Virtual
0          0    work kernel segment (lgpg_vsid=0)   L        16   16    0      16
ad70ad     d    work text or shared-lib code seg    m       893  171    0     893
fff8e6     3    work working storage                L         2    2    0       2
1eb19f2    1    work code                           m        78    0    0      78
a0a2b9     f    work working storage                m        37    0    0      37
43925a     2    work process private                m         4    1    0       4
df415      -    work                                s        40   12    0      40
1486151    -    clnt /dev/hd2:137980                s         1    0    -       -
89709a     -    clnt /dev/hd2:137896                s         1    0    -       -
```

- Note that ESID 3 is now a large segment (PSize=L).
- ALL large segments are added into the Large page counts above including the kernel!
- Ignore any kernel segments listed here.

Case Study of 4k pages vs 16M pages vmstat data

- 4K pages

us	sy	id	wa
3	14	64	18
3	14	65	18
4	15	61	21
3	18	58	21
4	18	56	23
4	20	54	22
4	19	54	22
3	17	59	22
3	17	59	21
3	16	62	19
3	15	63	18
2	13	65	20
3	14	63	20
3	16	62	19
3	15	63	19

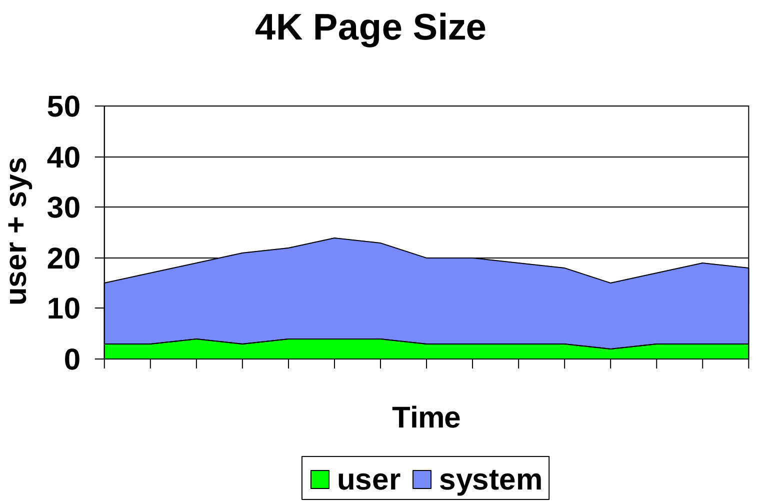
- 16M pages

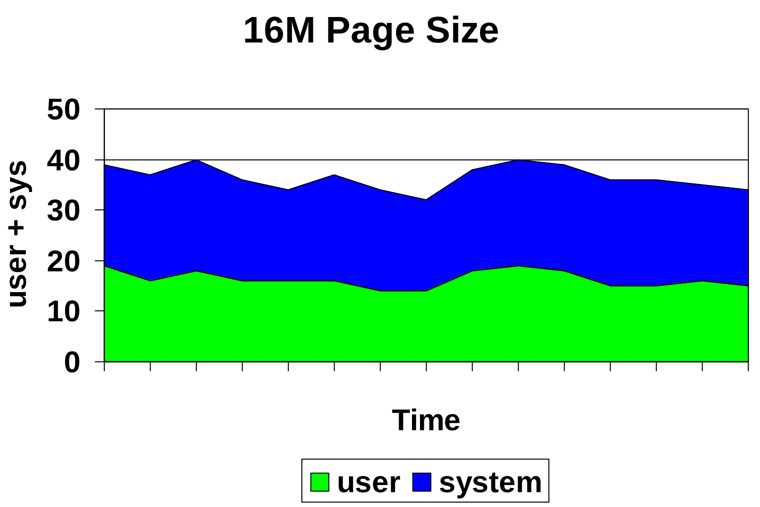
us	sy	id	wa
19	20	30	31
16	21	30	33
18	22	30	30
16	20	34	30
16	18	33	33
16	21	33	31
14	20	35	31
14	18	39	29
18	20	33	29
19	21	29	31
18	21	29	32
15	21	30	34
15	21	32	33
16	19	35	30
15	19	31	35

Note that although the system time is high in both cases, the user time is x greater with 16MB pages. This shows more real application work is being done.

Comparison of CPU utilization

Please note on the charts that although system time (in blue) is about the same in each of the 2 examples, the user time (in green) is significantly more when using 16MB pages. This clearly shows that the application is able to do more real work. This is the key to improving performance.





Tprof data with 16M pages

4K page size		16M page size	
Subroutine	%	Subroutine	%
==========	======	==========	======
h_cede_end_point	42.91	h_cede_end_point	40.81
.waitproc_find_run_queue	31.43	.waitproc_find_run_queue	29.07
.waitproc	14.40	.waitproc	13.86
.unlock_enable_mem	2.81	.unlock_enable_mem	2.07
h_put_tce_end_point	1.01	h_put_tce_end_point	1.50
.umem_move	0.51	.enable	0.44
.disable_lock	0.31	.umem_move	0.39
.enable	0.30	.trchook64	0.27

The small reduction in lock time may seem insignificant, but it has a significant affect on delays in TSM.

Splat data

4K page size

Lock Name, Class, or Address	Locks or Passes / CSec	Comb Spin
********************************	********	********
F1000F26E0023548	3489.768	0.5757
F1000500007B8980	8331.833	0.0701
F100050000061840	7052.326	0.0637
F1000A000173C288	318.401	0.0428
F10001004B608380	477.743	0.0149
F1000A0001734288	275.124	0.0126
F100010049C3E380	413.388	0.0082
F1000A2000851178	2645.290	0.0033
F1000A2000853178	2742.524	0.0021
F1000A2027E20288	55.081	0.0013

16M page size

Lock Name, Class, or Address	Locks or Passes / CSec	Comb Spin
********************************	********	********
F1000500002A6580	4641.372	0.0917
F1000500002D6B40	4416.202	0.0893
F1000A0001620288	507.605	0.0085
F1000A1800831178	5046.713	0.0083
F1000A1800833178	5078.881	0.0081
F1000F26980B36F8	23.330	0.0080
F10005000015DAA8	4906.026	0.0080
F1000A000168C288	312.481	0.0066
F1000A2000853178	3781.589	0.0053
0000000003B5F208	393.783	0.0053

Here we see that 16MB pages get more locks/passes per second and that the average spin times are much lower than when 4K pages are used.

Splat data – Hot functions

4K page size

Function Name	Acqui-sitions	Miss Rate	Spin Count
.pin_seg_range	75968	25.18	25569
.xlpin	6040	9.51	635
.as_geth	1578	12.58	227

ThreadID	Acqui-sitions	Miss Rate	Spin Count	Process Name
19661287	23465	32.47	11281	dsmserv
16974095	23465	33.32	11727	dsmserv
24052019	5590	13.49	872	dsmserv
25231547	2860	9.84	312	dsmserv
19661039	5525	8.63	522	dsmserv
24379413	13325	6.39	910	dsmserv
17039783	2925	4.91	151	dsmserv
7667765	3504	10.70	420	dsmserv
21037269	2927	7.46	236	dsmserv

16M page size

Function Name	Acqui-sitions	Miss Rate	Spin Count
.tcp_input0	11623	11.96	1579
.soereceive	12598	10.10	1416
.in_pcbhashlookup3	11623	6.40	795
.tcp_output	1912	8.87	186
.sosbwait	926	15.89	175
.soereceive	2734	5.10	147
.tcp_slowtimo	1	0.00	0
.tcp_fasttimo	3	0.00	0

ThreadID	Acqui-sitions	Miss Rate	Spin Count	Process Name
13762919	16	20.00	4	java
26148947	19565	8.95	1924	dsmserv
26017919	16	20.00	4	
24772947	30	16.67	6	
7995651	14	17.65	3	dsmserv
9371683	16	11.11	2	dsmserv
10748371	60	11.76	8	rm
24183189	18	21.74	5	smitty
18219479	34	17.07	7	dsmserv
15925517	18	10.00	2	dsmserv

The function experiencing lock contention is not the pin routines and we now see that dsmserv is not the only process taking the lock. This is because 16M pages are already pinned and there is not lock contention to pin them.

svmon data

4K page size

Pid	Command	Virtual	16MB
2687182	dsmserv	456048	N

PageSize		Virtual
s	4 KB	452512
m	64 KB	221
L	16 MB	0

16M page size

Pid	Command	Virtual	16MB
2753002	dsmserv	503117	Y

PageSize		Virtual
s	4 KB	12941
m	64 KB	428
L	16 MB	118

Note that we see 16M pages change from 'N' (not used) to 'Y'. M would indicates that the application has made 16M page usage Mandatory.

We also see that because 16M pages are in use there is corresponding reduction in 4k pages. This is what is expected.

Svmon page size changes

4K page size

Vsid	Esid	Type	PSize
1160816	14	work	s
10f080f	12	work	s
1110811	13	work	s
10b04cb	11	work	s
10f082f	15	work	s
10c03ec	16	work	s
10f0fef	17	work	s
1100ef0	18	work	s

16M page size

Vsid	Esid	Type	PSize
830bc3	16	work	L
8b050b	11	work	L
910b11	14	work	L
8a0dea	17	work	L
900b10	13	work	L
8f0b0f	12	work	L
920b12	15	work	L
9f6f7f	18	work	L

Note that the page size has changed from 4K pages(s) to 16M pages(L).

Data movement & CPU usage

- 4k pages - 5 seconds long - 2 readers, 5 writers

IOs Thread ID

630	16974095
96	17039783
435	19661039
609	19661287
562	24052019
321	24379413
130	25231547

- Each IO is 256K in size.
- Data moved = 2783 * 256K / 5sec = 139MB/sec

- 16MB pages - 2.7 seconds long 2 readers, 7 writers

855	17564041
202	19857621
840	21430571
205	22217143
822	22610359
154	24051857
87	24248823
130	27459717
788	6815957

- Each IO is 256K in size.
- Data moved = 4083 * 256k /2.7sec = 378MB/sec

Feedback from Customers who have implemented this change

- Overall CPU consumption for a given task is reduced.
- Since less time is spent spinning on locks, more real work gets done an processor utilization goes up.
- A number of TSM library operations are significantly faster (seconds instead of minutes).
- The number of tape drives that can be driven concurrently is increased.

Notes of Caution

- 16M pages are NOT pageable.
  - Using them incorrectly may drive your system out of memory and result in BAD things including a system crash.
  - First set LARGE_PAGE_DATA=Y to use 16MB pages when possible and 64KB if no 16MB pages are available.
  - Use 64K pages alone if 16M pages become a problem.
- Be sure to monitor the number of pages allocated and actually in use with the ‘svmon –G’ command.
- These changes will have less impact on systems with a smaller number of tape drives.

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.