Simulation Tech Tips

Model Solution Memory Tuning from a User's Perspective

Pete Ogilvie
Product Development Support

Model Solution and TVM

- When do you need to worry about this?
- Setting up your workstation
- Verifying memory available
- I-DEAS TVM Form
- Model Solution and TVM
  - Typical Model Solution memory warnings
  - Know issues with memory

When do you need to worry?

- You need to be aware of TVM settings with analyzing big models
- What is a big model???
  - **For a linear run, when number of degrees of freedom > 200,000**
    - DOF = Number of nodes x 6 for shell models
    - DOF = Number of nodes x 3 for solid models
- Contact and dynamics will require more memory than statics

Setting up your workstation

- RAM - Physical Memory
  - Get as much as you can afford
  - Most FE solves need 128 to 256 meg of Ram to run efficiently
  - 512 meg of RAM not uncommon
  - Multiple CPU machines may need more
- Swap - Virtual Memory
  - Cheaply extends machines memory by using disk space
  - Recommend setting Swap = 3 x Ram
  - Setting Swap greater than 3 x can be done but is inefficient
- TVM - Total Virtual Memory
  - I-DEAS Terminology
  - Tells I-DEAS what kind of resources your machine has
  - Directly tied to Swap resources of your workstation

Setting up your workstation

➤ SGI

Swap: swap -ln
RAM: hinv

Make sure the operating system (kernel parameters) can utilize all of the partitioned swap space.

The standard settings for an SGI workstation limit you to 512mb of virtual memory. This means that no matter what you set the memory preferences in I-DEAS, the total amount of virtual memory allocated for all partitions will be truncated to 512mb.

To increase your virtual memory to 1gb, log into root and enter the command "systune -i" and enter the following commands:

```
rlimit_vmem_max 0x40000000
y
rlimit_vmem_cur 0x40000000
y
rlimit_data_max 0x40000000
y
rlimit_data_cur 0x40000000
y
quit
```

For 1.5gb, use 0x60000000. For 2gb, use 0x7fffffff.

More information on kernel parameters can be obtained from the man pages: man systune.

Setting up your workstation

➤ **HP**

1) Find out how much swap is mounted. Use the command "swapinfo -tdfm". The last line will be the total, in megabytes (MB), amount of swap mounted on the machine.

2) The amount of swap that can be used by the kernel is defined as the Maximum Configurable Swap (MCS), where:

MCS = (maxswapchunks * swchunk * DEV_BSIZE) / (1024 * 1024)

where:

- DEV_BSIZE is equal to 1024
- swchunk is a kernel paramter that is 2048 by default and is generally not changed.
- maxswapchunks is a configurable kernel parameter.

You generally do not change DEV_BSIZE or swchunk, so this basically boils down to:

MCS = 2 * maxswapchunks (in MB)

So, the kernel parameter maxswapchunks needs to be set to at least one half of the swap value obtained from swapinfo (in step 1).

NOTE: The operating system (OS) will issue an error message during boot-up telling the user if maxswapchunks is not high enough, and telling him what value to increase it by.

Setting up your workstation

➤ **HP**

3) maxssiz should be equal to 64 MB (max is 79 MB).

4) maxtsiz should be equal to 64 MB. As I-DEAS executables exceed 64 MB this parameter may need to be set higher. For proper operation maxtsiz should be set equal to the size of the largest I-DEAS executable used (i.e. geomod.exe, suptab.exe, etc.).

5) maxdsiz = swap - maxssiz - maxtsiz
- where "swap" is the smaller of: a) mounted swap (step 1); or
b) max configurable swap (from step 2)

NOTE: maxdsiz, maxssiz, and maxtsiz are the maximum for each process. On multi-CPU systems, when attempting to run more than one I-DEAS executable at the same time, total available swap space restricts available virtual memory regardless of these kernel parameter settings.

NOTE: the current value of all kernel parameters can be determined either via SAM (as root) or "/usr/sbin/sysdef" (by anyone).

6) Total Virtual Memory (TVM) should be set to a value smaller than maxdsiz; leave a 30 MB buffer for OS overhead usage.

Setting up your workstation

➤ **Sun**

Swap: /usr/sbin/swap -l
RAM: /usr/sbin/prtconf

Make sure the operating system (kernel parameters) can utilize all of the partitioned swap space.

The kernal parameters of the sun need to be 16.2MB or larger. When setting up your kp you want to include the following line in the file /etc/system:

```
set shmsys:shminfo_shmmax=16986931.Reboot.
```

NOTE: This value needs to be increased if the user increases the Application Cache value beyond 15.4 on the Memory Usage Preferences form. This is the case when the Total Virtual Memory Allocation exceeds 200Mb using the default distribution percentages

(shmmax >= Application Cache + .8mb)

Setting up your workstation

➤ **IBM**

Swap: lsps -a
RAM: lsattr -El sys0 |grep (and then look for the line that says "realmem." This is a summation of total ram including system memory and L2 cache.) If this command doesn't work try the following:

```
lsdev -Cc memory
lsattr -El mem0 <- just for ram
lsattr -El L2cache0 <- just for L2 cache
```

There is a system limit file which can limit the amount of resources such as memory that a user may access. The file is /etc/security/limits and can only be read or modified by the root user.

```
default:
        fsize = -1
        core = 2048
        cpu = -1
        data = -1
        rss = -1
        stack = -1
```

The item of concern is the data item. By default this comes with a value of -1 which means unlimited. But if this has been changed, this would restrict the amount of virtual memory the user could access.

Setting up your workstation

➤ **IBM**

The code is also compiled with a parameter which limits the virtual memory to 768mb as recommended by IBM. To increase this limit one needs to change the header in the executable. To extend it to 1GB, issue the following command when in the directory where nbb.exe is located.

```
echo ’\0100\0\0\0’|dd of=nbb.exe bs=4 count=1 seek=19 conv=notrunc
```

For a larger value, increase the 0100 value as shown below

```
echo ’\0100\0\0\0’ -> 4 256MB segments (1    GB of data segment size)
echo ’\0120\0\0\0’ -> 5 256MB segments (1.25 GB of data segment size)
echo ’\0140\0\0\0’ -> 6 256MB segments (1.5  GB of data segment size)
echo ’\0160\0\0\0’ -> 7 256MB segments (1.75 GB of data segment size)
```

If it is increased to 2GB, there will be no space for shared libraries and you will probably crash. The 2GB limit is supposed to be increased in AIX4.2.

Verifying Memory Available

- MALLOC - *Memory Allocation*
  - **The Key to the problem - Get this right and problems go away**
  - **A programming call within I-DEAS that allocates (reserves) memory before crunching numbers**
  - **If your machine is configured properly, then you should be able to allocate most of your swap memory**
  - **The TVM settings tell I-DEAS how much memory it can MALLOC**
- **If MALLOC asks for more memory than the machine has available, then it will error out**
  - **Typically caused by three things**
    - TVM set to high
    - Swap - Kernel mismatch
    - Other processes on system also using memory

Verifying Memory Available

- I-DEAS is an expensive way to test ability to allocate memory
- Use a simple C routine to test, allocate then fill memory
- memtest param1 param2
  - param1 is max amount of memory to allocate (mb)
  - param2 is amount of memory (mb) it fills per iteration
- Compile with cc -Aa -o memtest memtest.c
  - use -N option on HP
- If memtest max allocation if less than you expect, you need to recheck system settings
- Don’t set I-deas TVM higher than max you can allocate

Verifying your settings

➤ memtest.c

```
#include "stdlib.h"
#include "stdio.h"
void main(int argc, char *argv[])
{
int i, i1, i2, inc, onemeg, imeg;
int count, rep;
int *p;
char ch;
onemeg = 1024*1024;
if(argc<3) {
  printf("no parameters entered - exiting\n");
  printf("Parameters: usemem [size in MB]
       [report interval]\n\n");
  exit(1);
}
imeg = atoi(argv[1]);
rep = atoi(argv[2]);
i1 = imeg*onemeg;
inc = sizeof(int);
i2 = i1/inc-1;
count = 0;
if(!(p=malloc(i1))) {
  printf("\nError allocating %d MB\n",imeg);
  exit(1);
}
printf("Successfully allocated %d MB memory\n\n",imeg);
for(i=0; i<=i2; i++) {
  p[i] = 1234567890;
  count++;
  if(count > rep*onemeg/inc) {
    printf("Have now used the first %d bytes
        (%d MB allocated)\n",i*inc/onemeg,imeg);
    printf("Hit enter to continue\n");
    ch = getchar();
    count = 0;
  }
}
printf("%d %d\n",i,i2);
free(p);
}
```

1 Line

1 Line

I-deas TVM Form

Application dynamics memory
-"On Demand" memory for applications
This is what Model Solution uses the most of

Display List
-Memory for graphics

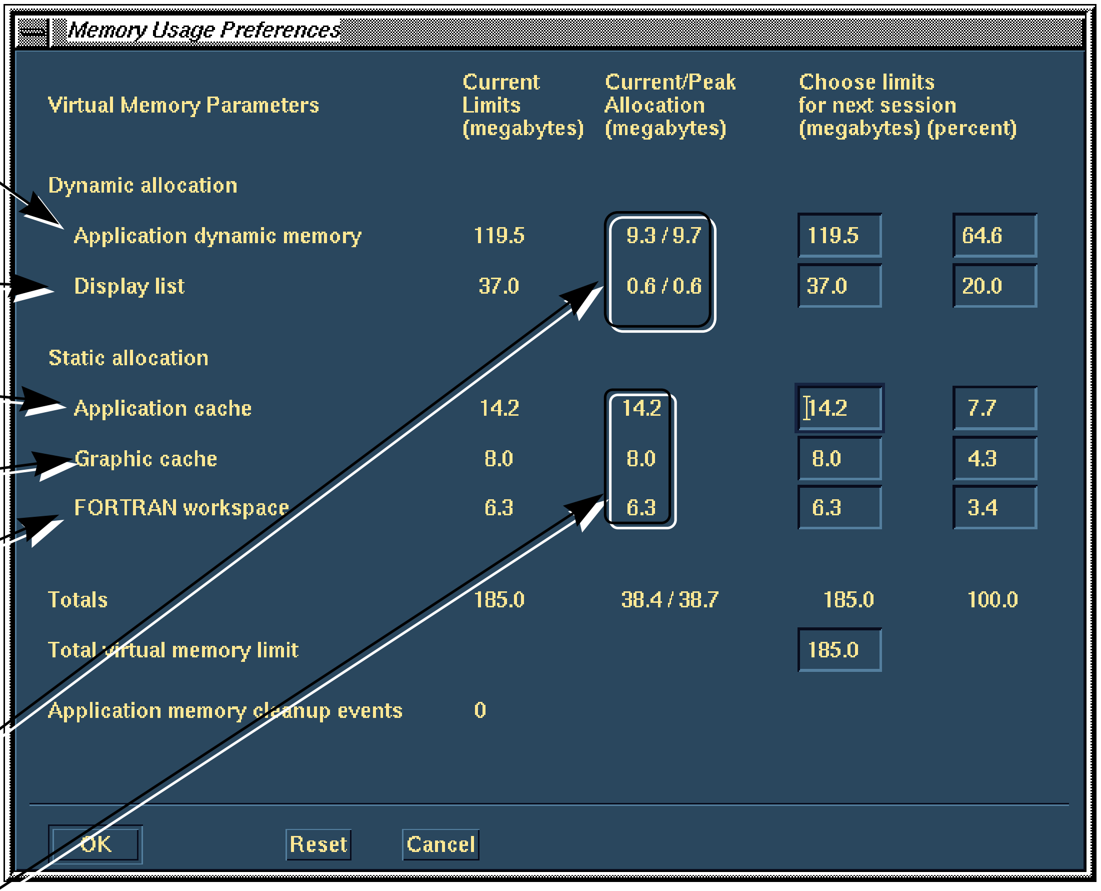
Application cache
-Keeps recently used commands in memory

Graphic cache
-Keeps recently used graphics in memory

FORTRAN workspace
-Storage for Fortran coding, Model Solution will use this

This is dynamic storage, amount used will change depending on program requirements

This is static storage, it will always be allocated when I-DEAS is started, amount used stays fixed



Memory Usage Preferences

Virtual Memory Parameters	Current Limits (megabytes)	Current/Peak Allocation (megabytes)	Choose limits for next session (megabytes)	(percent)
Dynamic allocation				
Application dynamic memory	119.5	9.3 / 9.7	119.5	64.6
Display list	37.0	0.6 / 0.6	37.0	20.0
Static allocation				
Application cache	14.2	14.2	14.2	7.7
Graphic cache	8.0	8.0	8.0	4.3
FORTRAN workspace	6.3	6.3	6.3	3.4
Totals	185.0	38.4 / 38.7	185.0	100.0
Total virtual memory limit			185.0	
Application memory cleanup events	0			

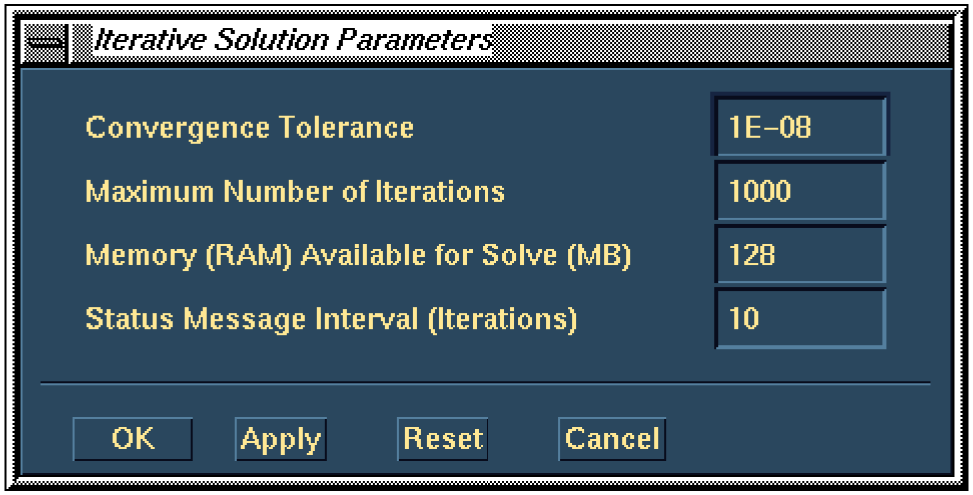
5% Min
2% Min
2% Min

I-deas TVM Form

- **Max TVM setting**
  - **No more than 90% of swap for most UNIX machines, have at least 50 mb more swap than TVM for operating system, etc**
    - For large models decreasing TVM may actually speed up the solve by forcing Model Solution to use scratch files (indicated by message “Sparse solve with be done out-of-memory” in list file)
  - **Less than maxdsiz-30 mb for HP**
  - **75% Total Paging (RAM+swap) file size for all drives for NT**
  - **This assumes you are the ONLY user on this machine**
- **TVM Percentages are saved in .user_param file in $SDRC_INSTL/team/master directory by default**
  - **You may want to write a script that sets TVM before session starts, this can then optimize settings for application and machine**

I-deas TVM Form

- TVM Percentages for Model Solution Sparse Matrix Solver
  - 75% Application Dynamic Memory
  - 5% Display List - Note: You may have to turn off dynamic viewing, autodraw when using this little
  - 5% Application Cache
  - 5% Graphic Cache
  - 10% Fortran Workspace
- Iterative Solver doesn't use Application Memory


Iterative Solution Parameters

Convergence Tolerance	1E-08
Maximum Number of Iterations	1000
Memory (RAM) Available for Solve (MB)	128
Status Message Interval (Iterations)	10

I-deas TVM Form

➤ Don't cheat

Scenario

Machine is verified to have 220 mb of swap (memory available)
You have default TVM settings

```
185 * 64.6% = 120   App Mem
185 *   20% =  37   Display List
185 *  7.7% =  14   App Cache          |
185 *  4.3% =   8   Graphics Cache     |  Static
185 *  3.4% =   6   Fortran w/s        |
```

You run Model Solution and get error saying

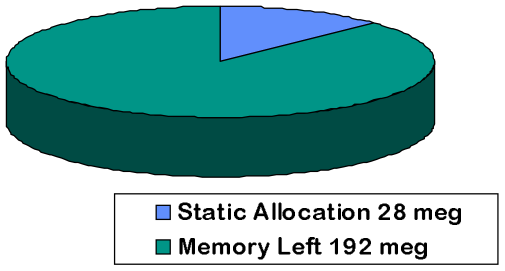
"E 21769 NEED MEMORY OF 200 MB BUT ONLY 108 IS AVAILABLE"

So you boost total TVM to 400, leaving percentages alone to get more Application Memory

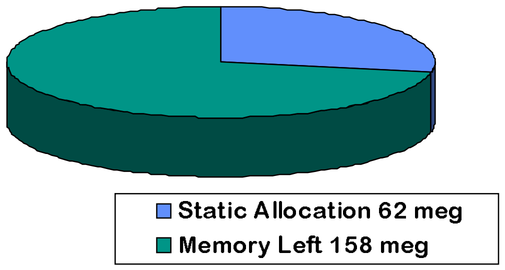
```
400 * 64.6% = 258   App Mem
400 *   20% =  80   Display List
400 *  7.7% =  31   App Cache          |
400 *  4.3% =  17   Graphics Cache     |  Static
400 *  3.4% =  14   Fortran w/s        |
```

Now you have also increased the amount of *static* allocation, as a result you have even less free memory

Total Pool 220 mb



Total Pool 220 mb



Model Solution and TVM

- Model Solution uses TVM settings to allocate memory for the sparse solver
- Model Solution will automatically allocate a fixed percentage of the Application Memory, depending on analysis type
  - For linear statics (>10k dof) it will attempt to allocate 90% of the App Mem
  - For contact it will attempt to allocate 60% of the App Mem
- If it cannot allocate full amount (machine refuses allocation request), it tries for a lesser amount until successful
  - If this amount is less than sparse solver requires then solution fails
- Model Solution is relatively polite, it will warn you if you get less than 80% Application Memory in a linear static solve
- Use param file entry to print estimates of required memory
  - sparse.memorymessage: 1

Model Solution and TVM

- Sparse solver has three modes of memory usage depending on problem size and amount of Application Memory available
- Messages appear in the *.lis file when running Model Solution
- "Sparse solve will be done in-memory"
  - Quickest, creates one scratch file *.rs0 and hypermatrix
  - Only seen with small models
- "Sparse solve will be done out-of-memory"
  - Creates scratch files, *.rs0,*.rs1,*.rs2 and hypermatrix file
- "Sparse solve will be done using the minimum memory option"
  - Creates scratch files, *.rs0,*.rs1,*.rs2 and hypermatrix file
  - Uses prodigious amounts of disk space, use caution

Model Solution and TVM

➤ **Examples of memory warnings and errors**

– **Normal run**

```
18:28:29 (CP      2.38     19.33) Boolean Formation Complete
18:28:29 (CP      0.14     19.47) Constraint Partitioning Complete
       TOTAL VIRTUAL MEMORY SPECIFIED               =  185 MB
       PERCENT ALLOCATED TO APPLICATION MEMORY      =   64 %
       MEMORY ALLOCATED FOR SPARSE MATRIX SOLVER    =  106 MB
       MEMORY USED BY SPARSE MATRIX SOLVER          =   48 MB
18:29:01 (CP     23.66     43.13) Sparse solve will be done out-of-memory
18:29:02 (CP      0.12     43.25) Est. decomp time = 407 cpu seconds
```

Linear Static run so,
Mem All = 185 * 64% * 90%

Memory used is based on problem size

Solve is done "out-of-memory" i.e. creates scratch files

Model Solution and TVM

➤ **Examples of memory warnings and errors**

– **Not enough Application Mem and not enough disk**

```
18:02:11 (CP     23.20    188.44) Boolean Formation Complete
18:02:12 (CP      0.95    189.39) Constraint Partitioning Complete
       TOTAL VIRTUAL MEMORY SPECIFIED             =  185 MB
       PERCENT ALLOCATED TO APPLICATION MEMORY    =   64 %
       MEMORY ALLOCATED FOR SPARSE MATRIX SOLVER  =  106 MB
 F  17997 APPLICATION MEMORY OF 106 MB IS INSUFFICIENT FOR THE SPARSE
          MATRIX SOLVER.  SIGNIFICANTLY INCREASE THE APPLICATION
          MEMORY IN THE MEMORY PREFERENCES FORM.  SEE SMARTVIEW
          FOR MORE INFORMATION
18:02:43 (CP      8.35    197.74) Sparse Matrix Setup Complete
```

This run fails for the obvious reason

Same model is run again with more Application Memory...

```
10:49:24 (CP     17.33    142.66) Boolean Formation Complete
10:49:26 (CP      0.70    143.36) Constraint Partitioning Complete
       TOTAL VIRTUAL MEMORY SPECIFIED             =  200 MB
       PERCENT ALLOCATED TO APPLICATION MEMORY    =   75 %
       MEMORY ALLOCATED FOR SPARSE MATRIX SOLVER  =  135 MB
10:51:59 (CP     23.42    166.78) Sparse solve will be done with minimum memory option
       MEMORY USED BY SPARSE MATRIX SOLVER        =  132 MB
10:55:43 (CP    161.36    328.14) Est. decomp time = 5706 cpu seconds
10:55:52 (CP      0.14    328.28) Sparse Matrix Setup Complete
10:55:55 (CP      0.10    328.38) Begin Decomposition
 E  21772 IO ERROR IN SOLVE   -505 - PROBABLY INSUFFICIENT DISK SPACE
          OR MEMORY SETTINGS ARE INCONSISTENT WITH THE VALUES
          OF THE KERNAL PARAMETERS OR SWAP SPACE
11:04:14 (CP    311.96    640.34) End Of Decomposition
```

Being very close to the edge, Model Solution is using the Minimuim Memory Option

Run fails again, but problem this time is *lack of disk space* for the hypermatrix and scratch files. *Important,* you can get this message if you have lots of free disk, but *incorrect system parameters*

Model Solution and TVM

➤ **Examples of memory warnings and errors**

– **Normal? run**

```
14:35:49 (CP      1.28     10.81) Boolean Formation Complete
14:35:49 (CP      0.14     10.95) Constraint Partitioning Complete
       TOTAL VIRTUAL MEMORY SPECIFIED                 =  600 MB
       PERCENT ALLOCATED TO APPLICATION MEMORY        =   75 %
       MEMORY ALLOCATED FOR SPARSE MATRIX SOLVER      =  213 MB
 W  21653 ONLY 213 MB OF MEMORY COULD BE ALLOCATED FOR THE SOLVE
          ALTHOUGH YOUR MEMORY PREFERENCES WERE SET HIGHER.  YOU
          MAY BE INCURRING A RESTRICTION SET BY THE OPERATING
          SYSTEM
       MEMORY USED BY SPARSE MATRIX SOLVER            =   65 MB
14:36:12 (CP     20.85     31.80) Sparse solve will be done in-memory
14:36:12 (CP      0.22     32.02) Est. decomp time = 136 cpu seconds
```

This run finished to completion without errors, but warning message indicates there was a problem with memory allocation. Model Solution will allocate less and less memory until it is successful. The amount it could allocate was greater than needed, therefore no error.

Small run (15000 DOF) Model Solution used the in-memory option

– **For HPs only**

```
*** FORTRAN I/O ERROR 913: OUT OF FREE SPACE
```

The message in your error.out file is usually symptomatic of kernal/swap/tvm mismatches, not lack of disk space or Fortran workspace

Model Solution and TVM

➤ **Examples of memory warnings and errors**

– **Contact**

```
10:01:40 (CP      0.00     26.39) Load Case : 1
10:01:40 (CP      0.00     26.40) Contact Iteration Number 1
10:01:40 (CP      0.01     26.41)
10:01:40 (CP      0.43     26.84) Form Contact Status
10:01:41 (CP      0.49     27.33) Number of contact status changes:     0
10:01:41 (CP      0.00     27.33) Number of inactive contacts:          441
10:01:41 (CP      0.01     27.34) Number of active open contactS:       0
10:01:41 (CP      0.00     27.34) Number of sticking contacts:          439
10:01:41 (CP      0.01     27.35) Number of sliding contacts:           0
10:01:46 (CP      3.45     30.80) Contact Stiffness Partitions Formed
       TOTAL VIRTUAL MEMORY SPECIFIED             = 1100 MB
       PERCENT ALLOCATED TO APPLICATION MEMORY     =   80 %
       MEMORY ALLOCATED FOR SPARSE MATRIX SOLVER   =  528 MB
       MEMORY USED BY SPARSE MATRIX SOLVER         =   91 MB
10:01:57 (CP      7.48     38.28) Sparse solve will be done in-memory
10:01:57 (CP      0.11     38.39) Est. decomp time = 72 cpu seconds
10:04:12 (CP      0.05    152.28)
10:04:12 (CP      0.01    152.29) Contact Iteration Number 2
10:04:13 (CP      0.01    152.30)
10:04:13 (CP      0.00    152.30) Form Contact Status
10:04:13 (CP      0.58    152.88) Number of contact status changes:     879
10:04:13 (CP      0.01    152.89) Number of inactive contacts:          289
10:04:13 (CP      0.01    152.90) Number of active open contactS:       0
10:04:13 (CP      0.01    152.91) Number of sticking contacts:          440
10:04:13 (CP      0.01    152.92) Number of sliding contacts:           151
10:04:21 (CP      5.83    158.75) Contact Stiffness Partitions Formed
       TOTAL VIRTUAL MEMORY SPECIFIED             = 1100 MB
       PERCENT ALLOCATED TO APPLICATION MEMORY     =   80 %
       MEMORY ALLOCATED FOR SPARSE MATRIX SOLVER   =  484 MB
 W  21653 ONLY 484 MB OF MEMORY COULD BE ALLOCATED FOR THE SOLVE
          ALTHOUGH YOUR MEMORY PREFERENCES WERE SET HIGHER.  YOU
          MAY BE INCURRING A RESTRICTION SET BY THE OPERATING
          SYSTEM
       MEMORY USED BY SPARSE MATRIX SOLVER         =  138 MB
10:04:32 (CP      7.79    166.54) Sparse solve will be done in-memory
10:04:32 (CP      0.11    166.65) Est. decomp time = 214 cpu seconds
```

Contact iteration 1 doesn't have a problem

Contact solution uses only 60% of the application memory for the sparse solver, so Mem Alloc = TVM * 80% * 60%

In this case Model Solution is at fault...
Seemingly minor memory leaks prevents the reallocation of the same amount of memory as requested in iteration 1
A fix is available...

Memory Used changes from iteration to iteration because number of active contacts change

Model Solution and TVM

- **Known issues**
  - **Contact Memory Loss (previous slide)**
    - Reduce TVM by 100-200 mb
    - Call Support Center or check WWW Tech Tips for prog file fix
    - Fixed in MS5 ptf
  - **SGI has 1.2 mb TVM limitation in MS4**
    - Fixed in MS5, MS4 may be fixed with O/S patch
  - **Sparse Solver scratch files can hit 2 gig UNIX file size limit**
    - Most UNIX platforms have optional Large File Systems, but I-deas does not take advantage of them (yet)
    - Typical error message (in startup window)

```
I-DEAS VMI Diagnostic (drfptr)
Illegal block number - exceeds maximum of 262143.
 ...called by drnwt ()
  LUN=30 INUM=262090 ILEN=2048 NREC=56
  process error reported by subroutine hdslco
  see hdslco abstract (ier = -505)
```

The error message shown above comes from the core IO routines when you attempt to write a file over 2GB in size (262143 records x 2048 words x 4 bytes/word = 2,147,475,456 and 2GB in decimal is 2,147,483,648).