

OpenVMS System Performance Management

Lab Exercises

Training



COMPAQ

OpenVMS System Performance Management

Lab Exercises



Training

COMPAQ

Notice

© 1999 Compaq Computer Corporation

COMPAQ, the Compaq logo, and the Digital logo are registered in the U.S. Patent and Trademark Office. Other product names mentioned herein may be the trademarks of their respective companies.

Confidential computer software. Valid license from Compaq or authorized sublicensor required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Compaq Computer Corporation shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is subject to change without notice.

Aero, ALPHA, ALPHA AXP, AlphaServer, AlphaStation, Armada, BackPak, COMPAQ, Compaq Insight Manager, CompaqCare logo, Counselor, DECterm, Deskpro, DIGITAL, DIGITAL logo, DIGITAL Alpha Systems, Digital Equipment Corporation, DIGITAL UNIX, DirectPlus, FASTART, Himalaya, InfoPak, Integrity, LicensePak, Ministation, NetFlex, NonStop, OpenVMS, PaqFax, Presario, ProLiant, ProLinea, ProSignia, QuickBack, QuickFind, Qvision, RDF, RemotePak, RomPak, ServerNet, SERVICenter, SmartQ, SmartStart, SmartStation, SolutionPak, SpeedPak, StorageWorks, Systempro/LT, Tandem, TechPak, TruCluster, and Tru64 UNIX are registered in United States Patent and Trademark Office.

Atalla, C-Series, Expand, FOX, Guardian, iTP, Measure, Netelligent, and PointView are trademarks of Compaq Computer Corporation.

Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Intel, Pentium, and Intel Inside are registered trademarks and Xeon is a trademark of Intel Corporation.

UNIX is a trademark in the US and other countries, licensed exclusively through X-Open Company Ltd.

OpenVMS System Performance Management

Lab Exercises

December 1999

Module 1 — Lab Exercises

Introduction	L1-1
Module 1 Labs (General Performance Concepts)	L1-1
Module 2 Labs (Layout of Virtual Address Space).....	L1-1
Module 3 Labs (Image Activation and Paging)	L1-1
Module 4 Labs (Interpreting Paging Metrics).....	L1-2
Module 5 Labs (Working Sets and Automatic Working Set Adjustment)	L1-2
Module 6 Labs (Locality and Paging Performance).....	L1-2
Module 7 Labs (Shared Image Costs and Benefits)	L1-3
Module 8 Labs (Modified Page List and Modified Page Writing)	L1-3
Module 9 Labs (Swapper and Memory Reclamation).....	L1-3
Module 10 Labs (System Faults).....	L1-4
Module 11 Labs (General I/O Flow)	L1-4
Module 12 Labs (Analyzing the Effects of Fragmentation)	L1-4
Module 13 Labs (Controller Optimizations).....	L1-4
Module 14 Labs (Files-11 ODS-2 Concepts)	L1-4
Module 15 Labs (Contiguity and the File System)	L1-4
Module 16 Labs (File System Caches).....	L1-5
Module 17 Labs (RMS Structures and Design Considerations)	L1-5
Module 18 Labs (RMS Utilities).....	L1-5
Module 19 Labs (RMS I/O Buffering Considerations)	L1-5
Module 20 Labs (\$MONITOR RMS)	L1-5
Module 21 Labs (Understanding CPU Performance).....	L1-6
Module 22 Labs (Understanding Time Spent in Modes)	L1-6
Module 23 Labs (OpenVMS Scheduling and Priorities).....	L1-6
Module 24 Labs (PIXSCAN and DORMANTWAIT).....	L1-6
Module 25 Labs (Locking Concepts)	L1-6
Module 26 Labs (Distributed Locking Concepts).....	L1-6
Module 27 Labs (MSCP Server).....	L1-6
Lab Support Code Source	L1-7
frag_pool.c	L1-7
loc1.c.....	L1-8
loc2.c.....	L1-8
stub.c.....	L1-8
big_code.mar	L1-9
big_code.com	L1-9
big_code_100.com.....	L1-9
touch_ppool.com	L1-9

fragger.c	L1-10
io_blaster.c	L1-12
sfrag.c.....	L1-14
rfrag.c.....	L1-16
SSFRAG.C.....	L1-18
create.com	L1-19
looper.c	L1-20
builder.com.....	L1-20
\$ @builder.....	L1-20
\$ show log perf\$	L1-20

Module 2 — Lab Solutions

Module 1 Labs (General Performance Concepts)	L2-1
Module 2 Labs (Layout of Virtual Address Space).....	L2-4
Module 3 Labs (Image Activation and Paging)	L2-10
Module 4 Labs (Interpreting Paging Metrics).....	L2-11
Module 5 Labs (Working Sets and Automatic Working Set Adjustment)	L2-12
Module 6 Labs (Locality and Paging Performance).....	L2-13
Module 7 Labs (Shared Image Costs and Benefits)	L2-24
Module 8 Labs (Modified Page List and Modified Page Writing)	L2-30
Module 9 Labs (The Swapper and Memory Reclamation).....	L2-31
Module 10 Labs (System Faults).....	L2-34
Module 11 Labs (General I/O Flow)	L2-36
Module 12 Labs (Analyzing the Effects of Fragmentation)	L2-37
Module 13 Labs (Controller Optimizations).....	L2-50
Module 14 Labs (Files-11 ODS-2 Concepts).....	L2-51
Module 15 Labs (Contiguity and the File System).....	L2-52
Module 16 Labs (File System Caches)	L2-55
Module 17 Labs (RMS Structures and Design Considerations)	L2-60
Module 18 Labs (RMS Utilities).....	L2-60
Module 19 Labs (RMS I/O Buffering Considerations)	L2-69
Module 20 Labs (\$MONITOR RMS)	L2-71
Module 21 Labs (Understanding CPU Performance).....	L2-73
Module 22 Labs (Understanding Time Spent in Modes).....	L2-73
Module 23 Labs (OpenVMS Scheduling and Priorities).....	L2-76
Module 24 Labs (PIXSCAN and DORMANTWAIT).....	L2-77
Module 25 Labs (Locking Concepts)	L2-78
Module 26 Labs (Distributed Locking Concepts).....	L2-78
Module 27 Labs (MSCP Server)	L2-78

Introduction

There are many lab exercises. You can pick and choose which labs are of most interest. It is also quite useful to explore other issues in lab. All programs and command procedures supporting the labs should be located in the PERF\$ directory.

Module 1 Labs (General Performance Concepts)

1. Use the SYSGEN utility to change the QUANTUM system parameter to 10 on the running system.
2. Use AUTOGEN to generate a feedback report. Examine the report.
3. Determine the hardware configuration of the system. Identify potential areas of bottleneck.

Module 2 Labs (Layout of Virtual Address Space)

1. Determine the size of the "Process Allocation Region".
2. Determine whether non-paged pool has expanded since the last boot.
3. Use SDA to determine the number of packets on the various "look-aside" lists.
4. Set the NPAG_INTERVAL to 2. Look for pool reclamation.
Run the program: PERF\$:FRAG_POOL. Examine the look-aside lists and determine size of the packets that have an increased number of free packets. Reduce the value for NPAG_GENTLE to 50 and examine the reclamation process. Approximately how long does it take for each reclamation pass? Why?
5. Determine the number of balance set slots in use and available. Compare these values against the number of process entries in use and available. Examine the SYSGEN parameters BALSETCNT and MAXPROCESSCNT.
6. Could "Artificial swapping" occur on this system?

Module 3 Labs (Image Activation and Paging)

1. Determine the "hard" and "soft" page fault rates on the system.
2. Determine the sizes of the Free and Modified Page lists.

Module 4 Labs (Interpreting Paging Metrics)

1. Of the page fault rates reported by \$MONITOR PAGE, which page faults can be attributed to image activation?
2. Which "page fault rates" might be reduced, through increasing working set list sizes?

Module 5 Labs (Working Sets and Automatic Working Set Adjustment)

1. Determine the following sizes of your working set list:
 - a. At Image Activation.
 - b. The amount of memory committed to your process.
 - c. The amount that your working set can grow to through loans.
2. Determine reasonable settings for working set defaults, quotas, and extents based on the activity on your system.
3. At what point will this system grant working set "loans"?

Module 6 Labs (Locality and Paging Performance)

1. Copy the files PERF\$:loc1.exe and PERF\$:loc2.exe to your directory. Determine the size of each image. Examine and compare the virtual memory layout of each image, using \$ANALYZE/IMAGE.

Now, enter the commands:

```
$SET WORK/LIMIT=1000/QUOTA=1000/EXTENT=1000
```

```
$SHOW STATUS
```

```
$RUN loc1
```

```
$SHOW STATUS
```

```
$SHOW STATUS
```

```
$RUN loc2
```

```
$SHOW STATUS
```

Compute the elapsed CPU time and number of page faults in each run of the programs.

Try it again with working set sizes of 8000, 16000, and 32000 pagelets.

How do they compare? Why are they different? (Examine the code.)

Module 7 Labs (Shared Image Costs and Benefits)

1. Copy the files PERF\$:big_code.exe and big_code.com and big_code_100 to your directory.
 - a. Issue the command \$ @big_code. Determine the total number of faults generated on the RUN of the program and the elapsed time for execution.
 - b. INSTALL BIG_CODE.EXE with the /SHARE qualifier. Repeat the method in step "a" twice. Compare the numbers. Explain the difference.
 - c. Determine whether BIG_CODE.EXE can be installed with the /RESIDENT qualifier. If not copy the object module BIG_CODE.OBJ and link it so that it can be installed /RESIDENT. Install it /RESIDENT and compare the numbers.
 - d. Run the command procedure BIG_CODE_100.COM with:
 - ♦ The image uninstalled.
 - ♦ The image installed /SHARE.
 - ♦ The image installed /RESIDENT
 - ♦ /RESIDENT and /SHARE=ADDRESS_DATA
 Compare execution times. Examine the paging statistics from \$MONITOR PAGE, while the command procedures are running.
2. Determine whether the image SYS\$SYSTEM:SDA.EXE may be installed /RESIDENT.
3. Install loc2 /SHARE and compare the paging statistics/execution time to when it is not installed. Do they change?

Module 8 Labs (Modified Page List and Modified Page Writing)

1. Determine the number of pages being written per Modified Page Write.
2. Determine the amount of space in use in the page file(s).

Module 9 Labs (Swapper and Memory Reclamation)

1. Determine whether any processes on the system are swapped.
2. Set FREEGOAL to the size of the free page list plus 1000. Determine whether any processes are swapped-out.
3. At what point will the system perform proactive memory reclamation?

Module 10 Labs (System Faults)

1. Determine the system fault rate.
2. Determine the size of the system working set list.
3. Issue the command `$@touch_ppool`. Determine the system fault rate while it is running.

Module 11 Labs (General I/O Flow)

1. Determine the Virtual I/O cache hit rate on the system.
2. Determine the I/O rates and queue depths for each disk.

Module 12 Labs (Analyzing the Effects of Fragmentation)

1. Run the program `PERF$:fragger` (use `$SHOW STATUS` to determine the Elapsed time of execution and I/Os generated by the program). Determine the:
 - a. Split transfer rate.
 - b. Window turn rate.
 - c. I/O rates to all disks.
 - d. The top direct I/O processes.
2. Determine which files `fragger` creates. Determine level of fragmentation in the files. Without changing the program can you get it to create files that are more contiguous.

Module 13 Labs (Controller Optimizations)

1. Use the program `PERF$:IO_BLASTER` to determine whether your controller/driver performs seek optimizations.

Module 14 Labs (Files-11 ODS-2 Concepts)

1. Determine the file open rates and the amount of time being used by the XQP.

Module 15 Labs (Contiguity and the File System)

1. Run the programs `PERF$:sfrag` and `PERF$:rfrag` on the files created by the `fragger` program. While they are running, compare the window turn and split transfer rates between the 2 programs. How do they compare? Why?

Module 16 Labs (File System Caches)

1. Determine the file system cache rates.
2. Issue the command:
`$SPAWN/NOWAIT DIR SYSS$SYSDEVICE:[000000...]/OUTPUT=NL:`
 Determine the file system cache rates. Repeat adding /FULL to DIR command.
3. Copy the command procedure PERF\$:CREATE.COM to your directory. Examine the command procedure. Invoke it using: @create/out=create.log. Examine create.log. Compare the file IDs and the logical block location of the files in the log. How do they compare? Why?
4. Sum up the values for ACP_DINDXCACHE, ACP_DIRCACHE, ACP_HDRCACHE, and ACP_MAPCACHE. Compare the sum against the value for "Maximum Buffers in FCP Cache" in the \$SHOW DEVICE/FULL display for a disk. How do they compare? Why?

Module 17 Labs (RMS Structures and Design Considerations)

This module has no formal labs.

Module 18 Labs (RMS Utilities)

1. Use EDIT/FDL to create an FDL file for an indexed file that will contain 10,000 initial records, with 100,000 additional records that are exactly 500 bytes in size, a primary key only that makes up the first 32 bytes. Use \$CREATE/FDL to create a "shell" container file to hold these records.
2. Use \$ANALYZE/RMS/FDL to create an FDL file for the SYSUAF.DAT file. Determine the number of index levels in the file.

Module 19 Labs (RMS I/O Buffering Considerations)

1. Determine the elapsed times and I/Os when running PERF\$:ssfrag to read the files generated by fragger. Determine the number of I/Os and the elapsed time required to run the program. Run the program with multi-block counts of 32, 64, and 96 blocks. Compare the elapsed times and I/O counts for the sample runs.

Module 20 Labs (\$MONITOR RMS)

1. Enable file statistics on the files X.DAT and Y.DAT. Determine the types of file operations performed on these files using the programs PERF\$:sfrag.exe and PERF\$:rfrag.exe.

Module 21 Labs (Understanding CPU Performance)

This module has no formal labs.

Module 22 Labs (Understanding Time Spent in Modes)

1. Determine where CPU time is being spent, when running the programs:
 - a. loc2 with a working set size of 8000
 - b. fragger
 - c. sfrag

Module 23 Labs (OpenVMS Scheduling and Priorities)

1. Run the program loc2 in the background 3 times with a default QUANTUM setting and working set sizes of 8000.

Do the same with QUANTUM set to 2. Do you notice any difference in response time?
2. With loc2 running what scheduling states do you see?

Module 24 Labs (PIXSCAN and DORMANTWAIT)

1. Run PERF\$:looper in the background at priority 1. Run it again in the background at priority 4. Using \$MONITOR PROCESS/TOPCPU determine whether the priority 1 process gets any CPU time. If so, why?

Module 25 Labs (Locking Concepts)

1. Determine the current number of locks and resources on the system.

Module 26 Labs (Distributed Locking Concepts)

Optional labs may be offered dependent on configuration.

Module 27 Labs (MSCP Server)

This module has no formal labs.

Lab Support Code Source

This code is included in source form for your interest. The command procedure builder.com will build all of the programs and set the PERFS: logical name to point to the current directory.

frag_pool.c

```

/* Program to allocate and free a bunch of Timer Queue Entries.
   Used to add packets to the 64 byte lookaside list.
   Author: Bruce Ellis, BRUDEN Corporation
*/
#include <starlet.h>
#include <ssdef.h>
#define check(STS) if(!((STS)&1)) sys$exit(STS)
#define TEN_MIN ((__int64)-10*60*1000*1000*10)
#define NUM_TIMERS 20
int main(void)
{
    __int64 exp_time = TEN_MIN;
    int i;
    int status;

    /* Set up 20 timers that would normally expire 10 minutes from now. */
    for(i=0;i<NUM_TIMERS;i++)
    {
        status = sys$setimr(i,&exp_time,0,0,i,0);
        check(status);
    }
    /* Cancel the timers. */
    for(i=0;i<NUM_TIMERS;i++)
    {
        status = sys$cantim(i,0);
        check(status);
    }
    return(SS$_NORMAL);
}

$

```

loc1.c

```
#include <stdlib.h>
#define NROWS 2000
#define NCOLS 2000
void stub(int);
int main(void)
{
    static int x[NROWS][NCOLS]={0};
    int i,j;

    for(i=0;i<NROWS;i++)
        for(j=0;j<NCOLS;j++)
        {
            x[i][j] += 42*i*j;
            stub(x[i][j]);
        }
    return(EXIT_SUCCESS);
}
$
```

loc2.c

```
#include <stdlib.h>
#define NROWS 2000
#define NCOLS 2000
void stub(int);
int main(void)
{
    static int x[NROWS][NCOLS]={0};
    int i,j;

    for(j=0;j<NCOLS;j++)
        for(i=0;i<NROWS;i++)
        {
            x[i][j] += 42*i*j;
            stub(x[i][j]);
        }
    return(EXIT_SUCCESS);
}
$
```

stub.c

```
void stub(int x)
{
}
$
```

big_code.mar

```

.psect code nowrt,shr,exe
.entry main,^m<>
.rept 20000
movl    #ss$_normal,r0
.endr
ret
.end    main

```

\$

big_code.com

```

$write sys$output "starting execution"
$show status
$run big_code
$show status
$write sys$output "ending execution"
$

```

big_code_100.com

```

$write sys$output "starting execution"
$show status
$x=1
$loop:
$run big_code
$x=x+1
$if x.le.100 then goto loop
$show status
$write sys$output "ending execution"
$

```

touch_ppool.com

```

$ana/sys
set output nla0:
examine @mmg$gl_pagedyn;@sgn$gl_pagedyn
exit
$
$

```

fragger.c

```
#include <stdlib.h>
#include <rms.h>
#include <starlet.h>
#include <rmsdef.h>
#include <fabdef.h>
#include <rabdef.h>
#include <ssdef.h>
#define check(STS) if(!((STS)&1)) sys$exit(STS)

#define REC_SIZE  512
#define MAX_REC   25000

int    main(void)
{
    int    status;
    struct FAB    x_fab = cc$rms_fab;
    int    i;
    struct RAB    x_rab = cc$rms_rab;
    struct FAB    y_fab = cc$rms_fab;
    struct RAB    y_rab = cc$rms_rab;
    char    record[REC_SIZE] = "Bruce Ellis was here!!!!!!!\nThat's right"
                                "Bruce Ellis of BRUDEN Corporation!";

    // Setup fab
    x_fab.fab$l_fna = "x.dat";
    x_fab.fab$b_fns = sizeof("x.dat");
    y_fab.fab$l_fna = "y.dat";
    y_fab.fab$b_fns = sizeof("y.dat");
    x_fab.fab$l_alq = y_fab.fab$l_alq = 1;
    x_fab.fab$b_fac = y_fab.fab$b_fac = FAB$M_PUT;
    x_fab.fab$b_fac = y_fab.fab$b_fac = FAB$M_GET|FAB$M_PUT;
    x_fab.fab$b_rfm=FAB$C_FIX;
    x_fab.fab$w_mrs=REC_SIZE;
    y_fab.fab$b_rfm=FAB$C_FIX;
    y_fab.fab$w_mrs=REC_SIZE;

    //// Setup rab
    x_rab.rab$l_fab = &x_fab;
    y_rab.rab$l_fab = &y_fab;
    x_rab.rab$l_rbf = record;
    x_rab.rab$w_rsz = sizeof(record);
    y_rab.rab$l_rbf = record;
    y_rab.rab$w_rsz = sizeof(record);
```

```
// Open the files
    status = sys$create(&x_fab);
    check(status);
    status = sys$create(&y_fab);
    check(status);
    status = sys$connect(&x_rab);
    check(status);
    status = sys$connect(&y_rab);
    check(status);

    for(i=0;i<MAX_REC;i++)
    {
        status = sys$put(&x_rab);
        check(status);
        status = sys$put(&y_rab);
        check(status);
    }

    return(EXIT_SUCCESS);
}

$
```

io_blaster.c

```
#include <stdlib.h>
#include <ssdef.h>
#include <stdio.h>
#include <starlet.h>
#include <iodef.h>
#include <iosbdef.h>
#include <descrip.h>
#include <time.h>
#define MAX_IOS 32
#define WAIT_MASK 0xFFFFFFFF
#define CLUSTER1 32
#define check(S) if(!((S)&1)) sys$exit(S)
#define BLOCK_SIZE 512
#include <dvidf.h>
#include <iledef.h>
typedef struct ctx_blocks
{
    struct _iosb iostat;
    int    which_io;
    int    which_lbn;
} CTX;

int    comp_ast(CTX *);
int    main(void)

{
    CTX    ctxblk[MAX_IOS];
    int    i;
    char    block[BLOCK_SIZE];
    int    chan;
    int    status;
    $DESCRIPTOR(disk,"SYS$SYSDEVICE:");
    int    max_lbn;
    ile3    dvi_list[] = {{sizeof(max_lbn),DVI$MAXBLOCK ,
                           0,0},{0}};

    srand((int)time(0));
    dvi_list[0].ile3$ps_bufaddr = &max_lbn;

    status = sys$assign(&disk,&chan,0,0,0);
    check(status);

    status = sys$getdviw(0,chan,0,dvi_list,0,0,0,0);
    check(status);
    for(i=0;i<MAX_IOS;i++)
    {
        ctxblk[i].which_io = i;
        status = sys$qio(CLUSTER1+i,chan,IO$READLBLK,
                        &ctxblk[i].iostat,comp_ast,ctxblk+i,
                        block,sizeof(block),
                        (ctxblk[i].which_lbn = rand()%max_lbn),0,0,0);
```

```
        check(status);
    }
    status = sys$wfland(CLUSTER1, WAIT_MASK);
    return(status);
}

int comp_ast(CTX *cb)
{
    check(cb->iostat.iosb$w_status);
    printf("Completed I/O %2d on block %10d\n", cb->which_io, cb->which_lbn);
    return(SS$_NORMAL );
}

$
```

sfrag.c

```
#include <stdlib.h>
#include <rms.h>
#include <starlet.h>
#include <rmsdef.h>
#include <fabdef.h>
#include <rabdef.h>
#include <ssdef.h>
#define check(sts)    if(!((sts)&1))          sys$exit(sts)

const  int    REC_SIZE = 512;
const  int    MAX_REC = 25000;
const  int    READ_LOOPS = 100000;
int     main(void)
{
    int     status;
    struct FAB    x_fab = cc$rms_fab;
    struct RAB    x_rab = cc$rms_rab;
    struct FAB    y_fab = cc$rms_fab;
    struct RAB    y_rab = cc$rms_rab;
    char    record[REC_SIZE];
    int     i,j;

    // Setup fab
    x_fab.fab$l_fna = "x.dat";
    x_fab.fab$b_fns = sizeof("x.dat");
    y_fab.fab$l_fna = "y.dat";
    y_fab.fab$b_fns = sizeof("y.dat");
    x_fab.fab$b_fac = y_fab.fab$b_fac = FAB$M_GET|FAB$M_PUT;
    x_fab.fab$b_rfm = y_fab.fab$b_rfm = FAB$C_FIX;
    //      x_fab.fab$l_fop = y_fab.fab$l_fop = FAB$M_CIF;
    // Setup rab
    x_rab.rab$l_fab = &x_fab;
    y_rab.rab$l_fab = &y_fab;
    x_rab.rab$l_ubf = record;
    x_rab.rab$w_usz = sizeof(record);
    y_rab.rab$l_ubf = record;
    y_rab.rab$w_usz = sizeof(record);

    for(j=0;j<READ_LOOPS;j++)
    {
```

```
// Open the files
    status = sys$open(&x_fab);
    check(status);
    status = sys$open(&y_fab);
    check(status);
    status = sys$connect(&x_rab);
    check(status);
    status = sys$connect(&y_rab);
    check(status);

    for(i=0;i<MAX_REC;i++)
    {
        status = sys$get(&x_rab);
        if(status != RMS$_EOF) check(status);
        status = sys$get(&y_rab);
        if(status != RMS$_EOF) check(status);
    }
    sys$close(&x_fab);

    sys$close(&y_fab);
}
return(EXIT_SUCCESS);
}

$
```

rfrag.c

```
#include <stdlib.h>
#include <time.h>
#include <rms.h>
#include <starlet.h>
#include <rmsdef.h>
#include <fabdef.h>
#include <rabdef.h>
#include <ssdef.h>
#define check(sts) if(!((sts)&1)) sys$exit(sts)

void    get_rec(int rec_num,struct RAB* x_rab,struct RAB* y_RAB);
const   int      REC_SIZE = 512;
const   int      MAX_REC = 25000;
const   int      READ_LOOPS = 100000;
int     main(void)
{
    int     status;
    struct FAB   x_fab = cc$rms_fab;
    struct RAB   x_rab = cc$rms_rab;
    struct FAB   y_fab = cc$rms_fab;
    struct RAB   y_rab = cc$rms_rab;
    char    record[REC_SIZE];
    int     block_no;
    int     i,j;

    // Setup fab
    x_fab.fab$l_fna = "x.dat";
    x_fab.fab$b_fns = sizeof("x.dat");
    y_fab.fab$l_fna = "y.dat";
    y_fab.fab$b_fns = sizeof("y.dat");
    x_fab.fab$b_fac = y_fab.fab$b_fac = FAB$M_GET|FAB$M_PUT;
    x_fab.fab$b_rfm=FAB$C_FIX;
    y_fab.fab$b_rfm=FAB$C_FIX;
    x_fab.fab$w_mrs=REC_SIZE;
    y_fab.fab$w_mrs=REC_SIZE;
    // Setup rab
    x_rab.rab$l_fab = &x_fab;
    y_rab.rab$l_fab = &y_fab;
    x_rab.rab$l_ubf = x_rab.rab$l_rbf = record;
    x_rab.rab$w_usz = x_rab.rab$w_rsz = REC_SIZE;
    y_rab.rab$l_ubf = y_rab.rab$l_rbf = record;
    y_rab.rab$w_usz = y_rab.rab$w_rsz = REC_SIZE;
    y_rab.rab$b_mbc = 6;
    x_rab.rab$b_mbc = 6;
    y_rab.rab$b_rac = RAB$C_RFA;
    x_rab.rab$b_rac = RAB$C_RFA;
    y_rab.rab$l_rop = RAB$C_RFA;
    x_rab.rab$l_rop = RAB$C_RFA;

    for(j=0;j<READ_LOOPS;j++)
    {
```

```

        // Open the files
        status = sys$open(&x_fab);
        check(status);
        status = sys$open(&y_fab);
        check(status);
        status = sys$connect(&x_rab);
        check(status);
        status = sys$connect(&y_rab);
        check(status);
        srand((int)clock());
        for(i=0;i<MAX_REC;i++)
        {
            get_rec(block_no=((rand())>>5)%MAX_REC)+1,&x_rab,&y_rab);
        }
        sys$close(&x_fab);

        sys$close(&y_fab);
    }
    return(EXIT_SUCCESS);
}
void get_rec(int rec_num,struct RAB* x_rab,struct RAB* y_rab)
{
    int status;
    y_rab->rab$l_rfa0 = rec_num;
    y_rab->rab$w_rfa4 = 0;
    x_rab->rab$l_rfa0 = rec_num;
    x_rab->rab$w_rfa4 = 0;
    status = sys$get(x_rab);
    if(status != RMS$_EOF) check(status);
    status = sys$get(y_rab);
    if(status != RMS$_EOF) check(status);
}
$

```

SSFRAG.C

```
#include <stdlib.h>
#include <rms.h>
#include <starlet.h>
#include <rmsdef.h>
#include <fabdef.h>
#include <rabdef.h>
#include <ssdef.h>

#define check(sts)      if(!((sts)&1)) sys$exit(sts)

const   int      REC_SIZE = 512;
const   int      MAX_REC = 25000;
const   int      READ_LOOPS = 2;
int      main(void)
{
    int      status;
    struct FAB    x_fab = cc$rms_fab;
    struct RAB    x_rab = cc$rms_rab;
    struct FAB    y_fab = cc$rms_fab;
    struct RAB    y_rab = cc$rms_rab;
    char      record[REC_SIZE];
    int      i,j;

    /* Setup fab */
    x_fab.fab$l_fna = "x.dat";
    x_fab.fab$b_fns = sizeof("x.dat");
    y_fab.fab$l_fna = "y.dat";
    y_fab.fab$b_fns = sizeof("y.dat");
    x_fab.fab$b_fac = y_fab.fab$b_fac = FAB$M_GET|FAB$M_PUT;
    x_fab.fab$b_rfm = y_fab.fab$b_rfm = FAB$C_FIX;

    /* Setup rab */
    x_rab.rab$l_fab = &x_fab;
    y_rab.rab$l_fab = &y_fab;
    x_rab.rab$l_ubf = record;
    x_rab.rab$w_usz = sizeof(record);
    y_rab.rab$l_ubf = record;
    y_rab.rab$w_usz = sizeof(record);
    for(j=0;j<READ_LOOPS;j++)
    {
```

```

        /* Open the files */
        status = sys$open(&x_fab);
        check(status);
        status = sys$open(&x_fab);
        check(status);
        status = sys$open(&y_fab);
        check(status);
        status = sys$connect(&x_rab);
        check(status);
        status = sys$connect(&y_rab);
        check(status);
        for(i=0;i<MAX_REC;i++)
        {
            status = sys$get(&x_rab);
            if(status != RMS$_EOF) check(status);
            status = sys$get(&y_rab);
            if(status != RMS$_EOF) check(status);
        }
        sys$close(&x_fab);
        sys$close(&y_fab);
    }
    return(EXIT_SUCCESS);
}

$

```

create.com

```

$ create zz.top
Three Guys with
long beards
$
$Dump/header/block=count:0 zz.top
$delete zz.top;
$ create zz.bottom
Missed the
charts
with this one
$
$Dump/header/block=count:0 zz.bottom
$
$
$
$

```

looper.c

```
int    main(void)
{
    while(1) {;}
}
```

builder.com

```
$SHOW TIME
$
$cc frag_pool
$link frag_pool
$
$cc loc1
$cc stub
$link loc1,stub
$
$cc loc2
$link loc2,stub
$
$mac/nooptimize big_code
$link big_code
$
$cc fragger
$link fragger
$
$cc sfrag
$link sfrag
$
$cc ssfrag
$link ssfrag
$
$cc rfrag
$link rfrag
$
$cc io_blaster
$link io_blaster
$cc looper
$link looper
$
$ define/system PERF$ 'f$environment("DEFAULT")'
$SHOW TIME
$
$
```

\$ @builder

```
29-NOV-1998 00:52:10
29-NOV-1998 00:55:32
$
```

\$ show log perf\$

```
"PERF$" = "SYS$SYSROOT:[SYSMGR]" (LNM$SYSTEM_TABLE)
$
```

Module 1 Labs (General Performance Concepts)

1. Use the SYSGEN utility to change the QUANTUM system parameter to 10 on the running system.

```
$ mcr sysgen
```

```
SYSGEN> SHOW QUANTUM
```

Parameter Name	Current	Default	Min.	Max.	Unit	Dynamic
QUANTUM	20	20	2	32767	10Ms	D

```
SYSGEN> SET QUANTUM 15
```

```
SYSGEN> WRITE ACTIVE
```

```
SYSGEN> EXIT
```

```
$ mcr sysman param show quantum
```

```
%SYSMAN-I-USEACTNOD, a USE ACTIVE has been defaulted on node ELLIS
```

```
Node ELLIS: Parameters in use: ACTIVE
```

Parameter Name	Current	Default	Minimum	Maximum	Unit	Dynamic
QUANTUM	15	20	2	32767	10Ms	D

```
$
```

2. Use AUTOGEN to generate a feedback report. Examine the report.

```
$ @sys$update:autoген savparams genparams

%AUTOGEN-I-BEGIN, SAVPARAMS phase is beginning.
%AUTOGEN-I-FEEDBACK, Gathering feedback
%AUTOGEN-I-NEWFILE, A new version of SYS$SYSTEM:AGEN$FEEDBACK.DAT
    has been created. You may wish to purge this file.
%AUTOGEN-I-END, SAVPARAMS phase has successfully completed.
%AUTOGEN-I-BEGIN, GETDATA phase is beginning.
%AUTOGEN-I-NEWFILE, A new version of SYS$SYSTEM:PARAMS.DAT has been created.
    You may wish to purge this file.
%AUTOGEN-I-END, GETDATA phase has successfully completed.
%AUTOGEN-I-BEGIN, GENPARAMS phase is beginning.
%AUTOGEN-I-NEWFILE, A new version of SYS$MANAGER:VMSIMAGES.DAT has been
created.
    You may wish to purge this file.
*****
%AUTOGEN-W-REPORT, Warnings were detected by AUTOGEN. Please review the
    information given in the file SYS$SYSTEM:AGEN$PARAMS.REPORT
*****
%AUTOGEN-I-REPORT, AUTOGEN has produced some informational messages which
    have been stored in the file SYS$SYSTEM:AGEN$PARAMS.REPORT. You may
    wish to review the information in that file.
%AUTOGEN-I-NEWFILE, A new version of SYS$SYSTEM:SETPARAMS.DAT has been
created.
    You may wish to purge this file.
%AUTOGEN-I-END, GENPARAMS phase has successfully completed.
$
$ type/page=save SYS$SYSTEM:AGEN$PARAMS.REPORT

AUTOGEN Parameter Calculation Report on node: ELLIS
This information was generated at 27-NOV-1998 22:28:06.09
AUTOGEN was run from SAVPARAMS to GENPARAMS - default execution specified

** No changes will be done by AUTOGEN **
    The values given in this report are what AUTOGEN would
    have set the parameters to.

Processing Parameter Data files
-----

Including parameters from: SYS$SYSTEM:MODPARAMS.DAT

Including parameters from: SYS$MANAGER:AGEN$NEW_NODE_DEFAULTS.DAT

The following problems were detected within MODPARAMS.DAT
    These problems should be reviewed immediately.

** WARNING ** - Duplicate value for VAXCLUSTER found.
    This overrides preceding requirements for this parameter in
    PARAMS.DAT
    Please review the parameter setting and if possible use MIN, MAX, or
    ADD
    ...
```

- Determine the hardware configuration of the system. Identify potential areas of bottleneck.



Note

\$SHOW MEMORY will describe memory limits. In \$ANALYZE/SYSTEM, the command CLUE CONFIG will show the CPU type.

```
$ mcr sysman io show bus
```

```
%SYSMAN-I-OUTPUT, command execution on node ELLIS
```

_Bus	Node	TR#	Name	Base	CSR
TURBO_SCSI	0	3	PMAZ-SCSI	FFFFFFFF8396E000	
COREIO	0	4	NI	FFFFFFFF8397A000	
COREIO	1	4	Serial line	FFFFFFFF83960000	
COREIO	2	4	Serial line	FFFFFFFF83960000	
COREIO	3	4	ISDN	FFFFFFFF83960000	
TURBOCHAN	6	2	PMAGB-BA	FFFFFFFF8397C000	

```
$
```

```
$ show dev d
```

Device	Device	Error	Volume	Free	Trans
Mnt					
Name	Status	Count	Label	Blocks	Count
Cnt					
ELLIS\$DKA200:	Mounted	0	AXP71FT2	454554	373 1
ELLIS\$DKA300:	Online	0			
ELLIS\$DKA500:	Online	0			
ELLIS\$DKA600:	Online wrtlck	0			

```
$
```

Module 2 Labs (Layout of Virtual Address Space)

1. Determine the size of the "Process Allocation Region".

```
$ mcr sysgen show ctlpages
Parameter Name      Current      Default      Min.      Max.      Unit
Dynamic
-----
CTLPPAGES           100         100         20        -1 Pagelets
$
```

The following command will show the process allocation region being used by a process:

```
$ sh proc/mem

27-NOV-1998 22:41:57.29  User: SYSTEM      Process ID:  20200127
                        Node: ELLIS        Process name: "_FTA1:"

Process Dynamic Memory Area
  Current Size (bytes)      57344    Current Size (pagelets)      112
  Free Space (bytes)       41084    Space in Use (bytes)        16260
  Largest Variable Block   40236    Smallest Variable Block      16
  Number of Free Blocks     12      Free Blocks LEQU 64 Bytes     8
$
$
```

2. Determine whether nonpaged pool has expanded since the last boot.

\$ show memory/pool/full

System Memory Resources on 27-NOV-1998 22:43:17.52

Nonpaged Dynamic Memory (Lists + Variable)			
Current Size (bytes)	3293184	Current Size (pagelets)	6432
Initial Size	2048000	Initial Size (pagelets)	4000
Maximum Size	10240000	Maximum Size (pagelets)	20000
Free Space (bytes)	77824	Space in Use (bytes)	3215360
Largest Variable Block	3072	Smallest Variable Block	64
Number of Free Blocks	92	Free Blocks LEQU 64 Bytes	2
Free Blocks on Lookasides	52	Lookaside Space (bytes)	54912

(Bus Addressable Memory requests use Nonpaged Dynamic Memory)

Paged Dynamic Memory			
Current Size (PAGEDYN)	1835008	Current Size (pagelets)	3584
Free Space (bytes)	965520	Space in Use (bytes)	869488
Largest Variable Block	963840	Smallest Variable Block	16
Number of Free Blocks	34	Free Blocks LEQU 64 Bytes	28

Lock Manager Dynamic Memory			
Current Size (Mbytes)	1.00	Current Size (pages)	128
Free Space (Mbytes)	0.37	Hits	2730
Space in Use (Mbytes)	0.62	Misses	0
Number of Empty Pages	48	Expansions	128
Number of Free Packets	1529	Packet Size (bytes)	256

\$

Non-paged pool has expanded since the "current size" (3293184) is greater than the initial size (2048000).

3. Use SDA to determine the number of packets on the various "look-aside" lists.

\$ **analyze/system**

OpenVMS (TM) Alpha system analyzer

SDA> **clue mem/look**

Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:

```
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid, 5 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid, 2 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid, 4 elements
Listhead Addr: FFFFFFFF.80C34298   Size:  256   Status: Valid, 3 elements
Listhead Addr: FFFFFFFF.80C342A0   Size:  320   Status: Valid, 2 elements
Listhead Addr: FFFFFFFF.80C342A8   Size:  384   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C342B0   Size:  448   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C342B8   Size:  512   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C342C0   Size:  576   Status: Valid, 5 elements
Listhead Addr: FFFFFFFF.80C342C8   Size:  640   Status: Valid, 2 elements
Listhead Addr: FFFFFFFF.80C342D0   Size:  704   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C342D8   Size:  768   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C342E0   Size:  832   Status: Valid, 2 elements
Listhead Addr: FFFFFFFF.80C342E8   Size:  896   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C342F0   Size:  960   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C342F8   Size: 1024   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C34300   Size: 1088   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C34308   Size: 1152   Status: Valid, 1 element
Listhead Addr: FFFFFFFF.80C34310   Size: 1216   Status: Valid, 1 element
...
Listhead Addr: FFFFFFFF.80C34648   Size: 7808   Status: Valid, empty
Listhead Addr: FFFFFFFF.80C34650   Size: 7872   Status: Valid, empty
Listhead Addr: FFFFFFFF.80C34658   Size: 7936   Status: Valid, empty
Listhead Addr: FFFFFFFF.80C34660   Size: 8000   Status: Valid, empty
Listhead Addr: FFFFFFFF.80C34668   Size: 8064   Status: Valid, empty
Listhead Addr: FFFFFFFF.80C34670   Size: 8128   Status: Valid, empty
Listhead Addr: FFFFFFFF.80C34678   Size: 8192   Status: Valid, empty
```

Total free space: 0000D640 (hex) 54848 (dec) bytes

SDA>

4. Set the NPAG_INTERVAL to 2. Look for pool reclamation.

Run the program: PERF\$:FRAG_POOL. Examine the look-aside lists and determine size of the packets that have an increased number of free packets. Reduce the value for NPAG_GENTLE to 50 and examine the reclamation process. Approximately how long does it take for each reclamation pass? Why?

Each reclamation pass touches 2 lists. There are 128 lists. With a default setting of 30 seconds per pass, it would take 1920 seconds (or 32 minutes) to reclaim all lists. With NPAG_INTERVAL set to 2, it would take 128 seconds (or 2 minutes and 8 seconds) to reclaim all lists. The cost for this reclamation would be additional Interrupt State/Stack time.

```
$ r frag_pool
$
$ ana/sys
```

OpenVMS (TM) Alpha system analyzer

```
SDA> clue mem/look
```

Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:

```
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid, 20 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid,  2 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid, 14 elements
Listhead Addr: FFFFFFFF.80C34298   Size:  256   Status: Valid,  6 elements
Listhead Addr: FFFFFFFF.80C342A0   Size:  320   Status: Valid,  8 elements
Listhead Addr: FFFFFFFF.80C342A8   Size:  384   Status: Valid,  1 element
...
```

```
SDA> clue mem/look
```

Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:

```
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid, 19 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid,  2 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid, 14 elements
Listhead Addr: FFFFFFFF.80C34298   Size:  256   Status: Valid,  6 elements
...
```

```
SDA> clue mem/look
```

Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:

```
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid, 18 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid,  2 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid, 14 elements
...
```

```
SDA> clue mem/look
```

Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:

```
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid, 18 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid,  2 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid, 14 elements
...
```

```
SDA> spawn
```

```
$ mcr sysgen
SYSGEN> SET NPAG_INT 2
SYSGEN> SET NPAG_GENTLE 50
SYSGEN> WRITE ACT
SYSGEN> EXIT
$
$ lo
Process SYSTEM_1 logged out at 27-NOV-1998 23:02:27.75
SDA>
SDA> clue mem/look
Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid, 15 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid,  7 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid, 14 elements
Listhead Addr: FFFFFFFF.80C34298   Size:  256   Status: Valid,  6 elements
...
SDA> clue mem/look
Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid,  8 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid,  3 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid,  7 elements
Listhead Addr: FFFFFFFF.80C34298   Size:  256   Status: Valid,  3 elements
...
SDA> clue mem/look
Non-Paged Dynamic Storage Pool - Lookaside List Queue Information:
-----
Listhead Addr: FFFFFFFF.80C34280   Size:   64   Status: Valid,  5 elements
Listhead Addr: FFFFFFFF.80C34288   Size:  128   Status: Valid,  2 elements
Listhead Addr: FFFFFFFF.80C34290   Size:  192   Status: Valid,  4 elements
Listhead Addr: FFFFFFFF.80C34298   Size:  256   Status: Valid,  1 element
```

5. Determine the number of balance set slots in use and available. Compare these values against the number of process entries in use and available. Examine the SYSGEN parameters BALSETCNT and MAXPROCESSCNT.

```
$ show memory/slots
System Memory Resources on 27-NOV-1998 23:18:35.76

Slot Usage (slots):          Total      Free      Resident      Swapped
  Process Entry Slots          136       106         30         0
  Balance Set Slots           134       106         28         0
$
$ mcr sysgen
SYSGEN> SH BALSETCNT
Parameter Name      Current    Default    Min.      Max.      Unit
Dynamic
-----
BALSETCNT           134        30         8        16384 Slots
SYSGEN> SHOW MAXPROCESSCNT
Parameter Name      Current    Default    Min.      Max.      Unit
Dynamic
-----
MAXPROCESSCNT       136        32        12        16384 Processes
SYSGEN> EXIT
$
```

6. Could "Artificial swapping" occur on this system?

No. The setting for BALSETCNT (== MAXPROCESSCNT-2) allows all possible processes to hold a balance slot.

Module 3 Labs (Image Activation and Paging)

1. Determine the "hard" and "soft" page fault rates on the system.

The "Hard fault rate" is the "Page Read I/O Rate" in **\$MONITOR PAGE**.

The soft fault rate would be the "Page Fault Rate" minus the "Page Read I/O Rate".

\$ monitor page

```
OpenVMS Monitor Utility
PAGE MANAGEMENT STATISTICS
on node ELLIS
27-NOV-1998 23:21:30.15
```

	CUR	AVE	MIN	MAX
Page Fault Rate	0.00	0.41	0.00	1.00
Page Read Rate	0.00	0.33	0.00	1.33
Page Read I/O Rate	0.00	0.16	0.00	0.66

...

2. Determine the sizes of the Free and Modified Page lists.

This information could also be obtained through **\$MONITOR PAGE**.

\$ show memory/physical

System Memory Resources on 27-NOV-1998 23:22:05.83

Physical Memory Usage (pages):	Total	Free	In Use	Modified
Main Memory (160.00Mb)	20480	10907	8448	1125

Of the physical pages in use, 3579 pages are permanently allocated to OpenVMS.

\$

Module 4 Labs (Interpreting Paging Metrics)

1. Of the page fault rates reported by \$MONITOR PAGE, which page faults can be attributed to image activation?

The following types of faults are most commonly attributed to image activation (pieces of the \$MONITOR PAGE display):

CUR	AVE	MIN	MAX				
Page Read I/O Rate			0.00	0.05	0.00	0.66	
Free List Fault Rate			0.00	0.00	0.00	0.00	
Demand Zero Fault Rate			0.00	0.02	0.00	0.33	
Global Valid Fault Rate			0.00	0.05	0.00	0.66	

2. Which "page fault rates" might be reduced, through increasing working set list sizes?

Modified List Faults and possibly Free List, and Global Valid Faults.

Module 5 Labs (Working Sets and Automatic Working Set Adjustment)

1. Determine the following sizes of your working set list:
 - a. At Image Activation.
 - b. The amount of memory committed to your process.
 - c. The amount that your working set can grow to through loans.
 - ♦ Limit
 - ♦ Quota
 - ♦ Extent

```
$ show working_set
Working Set (pagelets) /Limit=1424 /Quota=2848 /Extent=73728
Adjustment enabled      Authorized Quota=2848  Authorized Extent=73728

Working Set (8Kb pages) /Limit=89 /Quota=178 /Extent=4608
                        Authorized Quota=178  Authorized Extent=4608
$
```

2. Determine reasonable settings for working set defaults, quotas, and extents based on the activity on your system.

Based on the following information, reasonable settings (rounding 582 up to 600) might be:

- Limit == 600 pages * 16 pagelets/page == 9600 pagelets
- Quota == 600 pages * 16 pagelets/page == 9600 pagelets
- Extent == 1800 pages * 16 pagelets/page == 28800 pagelets

```
$ ana/sys
```

OpenVMS (TM) Alpha system analyzer

```
SDA> ev @pfn$gl_phypgcnt/@pms$gl_proccntmax
Hex = 00000000.00000246   Decimal = 582      BUG$_NETNOBUF+000006
SDA>
```

3. At what point will this system grant working set "loans"?

```
$ mcr sysgen show borrowlim
Parameter Name      Current   Default   Min.      Max.      Unit
-----
Dynamic
-----
BORROWLIM          154       300        0         -1 Pages
D
$
```

Module 6 Labs (Locality and Paging Performance)

1. Copy the files PERF\$:loc1.exe and PERF\$:loc2.exe to your directory. Determine the size of each image. Examine and compare the virtual memory layout of each image, using \$ANALYZE/IMAGE.

Now, enter the commands:

```
$SET WORK/LIMIT=1000/QUOTA=1000/EXTENT=1000
```

```
$SHOW STATUS
```

```
$RUN loc1
```

```
$SHOW STATUS
```

```
$SHOW STATUS
```

```
$RUN loc2
```

```
$SHOW STATUS
```

Compute the elapsed CPU time and number of page faults in each run of the programs.

Try it again with working set sizes of 8000, 16000, and 32000 pagelets. How do they compare? Why are they different? (Examine the code.)

Note, the images are the same size on disk.

```
$ dir/size loc1.exe,, loc2.exe;
```

```
Directory SYS$SYSROOT:[SYSMGR]
```

```
LOC1.EXE;6          7
LOC2.EXE;3          7
```

```
Total of 2 files, 14 blocks.
```

```
$
```

\$ANALYZE/IMAGE shows that the images have the same layout in virtual memory.

```
$ more="type/page=save sys$input"
```

```
$ pipe ana/image loc1 | more
```

```
Analyze Image
```

```
27-NOV-1998 23:30:38.44 Page 1
```

```
SYS$SYSROOT:[SYSMGR]LOC1.EXE;6
```

```
ANALYZ A07-04
```

```
This is an OpenVMS Alpha image file
```

```
IMAGE HEADER
```

Fixed Header Information

```
image format major id: 3, minor id: 0
header block count: 2
image type: executable (EIHD$K_EXE)
I/O channel count: default
I/O pagelet count: default
Symbol Vector Virtual Address: %X'00000000'
Symbol Vector Size: 0 bytes
Virtual Memory Block Size: 65536 (BPAGE = 16)
Fixup Section Virtual Address: %X'00F80000'
linker flags:
  (0) EIHD$V_LNKDEBUG  0
  (1) EIHD$V_LNKNOTFR  0
```

```
...
```

```
1) image section descriptor (36 bytes)
  byte count: 512
  base virtual address: %X'00010000' (P0 space)
  page fault cluster size: default
  ISD flags:
    (0) EISD$V_GBL      0
    (1) EISD$V_CRF      1
    (2) EISD$V_DZRO     0
    (3) EISD$V_WRT      1
```

```
(4) EISD$V_INITALCOD 0
(5) EISD$V_BASED     0
(6) EISD$V_FIXUPVEC  0
(7) EISD$V_RESIDENT  0
(8) EISD$V_VECTOR    0
(9) EISD$V_PROTECT   0
(10) EISD$V_LASTCLU  1
(11) EISD$V_EXE      0
(12) EISD$V_NONSHRADR 1
section type: EISD$K_NORMAL
base VBN: 3
```

```
2) image section descriptor (36 bytes)
   byte count: 16000000
   base virtual address: %X'00020000' (P0 space)
   page fault cluster size: default
   ISD flags:
       (0) EISD$V_GBL      0
       (1) EISD$V_CRF      0
       (2) EISD$V_DZRO     1
       (3) EISD$V_WRT      1
       (4) EISD$V_INITALCOD 0
       (5) EISD$V_BASED     0
       (6) EISD$V_FIXUPVEC  0
       (7) EISD$V_RESIDENT  0
       (8) EISD$V_VECTOR    0
       (9) EISD$V_PROTECT   0
       (10) EISD$V_LASTCLU  1
       (11) EISD$V_EXE      0
       (12) EISD$V_NONSHRADR 0
   section type: EISD$K_NORMAL
```

```
3) image section descriptor (36 bytes)
   byte count: 512
   base virtual address: %X'00F70000' (P0 space)
   page fault cluster size: default
   ISD flags:
       (0) EISD$V_GBL      0
       (1) EISD$V_CRF      0
       (2) EISD$V_DZRO     0
       (3) EISD$V_WRT      0
       (4) EISD$V_INITALCOD 0
       (5) EISD$V_BASED     0
       (6) EISD$V_FIXUPVEC  0
       (7) EISD$V_RESIDENT  0
       (8) EISD$V_VECTOR    0
       (9) EISD$V_PROTECT   0
       (10) EISD$V_LASTCLU  1
       (11) EISD$V_EXE      1
       (12) EISD$V_NONSHRADR 0
   section type: EISD$K_NORMAL
   base VBN: 4
```

- 4) image section descriptor (36 bytes)
byte count: 512
base virtual address: %X'00F80000' (P0 space)
page fault cluster size: default
ISD flags:
 (0) EISD\$V_GBL 0
 (1) EISD\$V_CRF 1
 (2) EISD\$V_DZRO 0
 (3) EISD\$V_WRT 1
 (4) EISD\$V_INITALCOD 0
 (5) EISD\$V_BASED 0
 (6) EISD\$V_FIXUPVEC 1
 (7) EISD\$V_RESIDENT 0
 (8) EISD\$V_VECTOR 0
 (9) EISD\$V_PROTECT 0
 (10) EISD\$V_LASTCLU 0
 (11) EISD\$V_EXE 0
 (12) EISD\$V_NONSHRADR 0
section type: EISD\$K_NORMAL
base VBN: 5
- 5) image section descriptor (36 bytes)
byte count: 10240
base virtual address: %X'7FFF0000' (P1 space)
page fault cluster size: default
ISD flags:
 (0) EISD\$V_GBL 0
 (1) EISD\$V_CRF 0
 (2) EISD\$V_DZRO 1
 (3) EISD\$V_WRT 1
 (4) EISD\$V_INITALCOD 0
 (5) EISD\$V_BASED 0
 (6) EISD\$V_FIXUPVEC 0
 (7) EISD\$V_RESIDENT 0
 (8) EISD\$V_VECTOR 0
 (9) EISD\$V_PROTECT 0
 (10) EISD\$V_LASTCLU 1
 (11) EISD\$V_EXE 0
 (12) EISD\$V_NONSHRADR 0
section type: EISD\$K_USRSTACK

```

6) image section descriptor (56 bytes)
   byte count: 1521152
   base virtual address: %X'00000000' (P0 space)
   page fault cluster size: default
   ISD flags:
       (0) EISD$V_GBL      1
       (1) EISD$V_CRF      0
       (2) EISD$V_DZRO     0
       (3) EISD$V_WRT      0
       (4) EISD$V_INITALCOD 0
       (5) EISD$V_BASED     0
       (6) EISD$V_FIXUPVEC  0
       (7) EISD$V_RESIDENT  0
       (8) EISD$V_VECTOR    0
       (9) EISD$V_PROTECT   0
      (10) EISD$V_LASTCLU   0
      (11) EISD$V_EXE       0
      (12) EISD$V_NONSHRADR 0
   section type: EISD$K_SHRPIC
   base VBN: 0
   global section major id: %X'01', minor id: %X'000001'
   match control: ISD$K_MATLEQ
   global section name: "DECC$SHR_001"
7) image section descriptor (64 bytes)
   byte count: 8192
   base virtual address: %X'00000000' (P0 space)
   page fault cluster size: default
   ISD flags:
       (0) EISD$V_GBL      1
       (1) EISD$V_CRF      0
       (2) EISD$V_DZRO     0
       (3) EISD$V_WRT      0
       (4) EISD$V_INITALCOD 0
       (5) EISD$V_BASED     0
       (6) EISD$V_FIXUPVEC  0
       (7) EISD$V_RESIDENT  0
       (8) EISD$V_VECTOR    0
       (9) EISD$V_PROTECT   0
      (10) EISD$V_LASTCLU   0
      (11) EISD$V_EXE       0
      (11) EISD$V_EXE       0
      (12) EISD$V_NONSHRADR 0
   section type: EISD$K_PRVFXD
   base VBN: 0
   global section major id: %X'43', minor id: %X'3A2DAD'
   match control: ISD$K_MATEQU
   global section name: "SYS$PUBLIC_VECTORS_001"

```

...

```
$ ana/image loc1 /out=loc1.ana
%ANALYZE-I-ERRORS, SYS$SYSROOT:[SYSMGR]LOC1.EXE;6          0 errors
$ ana/image loc2 /out=loc2.ana
%ANALYZE-I-ERRORS, SYS$SYSROOT:[SYSMGR]LOC2.EXE;3          0 errors
$ diff loc1.ana loc2.ana
*****
File SYS$SYSROOT:[SYSMGR]LOC1.ANA;1
   2   Analyze Image                                27-NOV-1998 23:36:16.66   P
   3   SYS$SYSROOT:[SYSMGR]LOC1.EXE;6
   4   ANALYZ A07-04
*****
File SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
   2   Analyze Image                                27-NOV-1998 23:36:23.29   P
   3   SYS$SYSROOT:[SYSMGR]LOC2.EXE;3
   4   ANALYZ A07-04
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC1.ANA;1
   51                                image name: "LOC1"
   52                                image file identification: "V1.0"
   53                                image file build identification: ""
   54                                link date/time: 23-NOV-1998 15:36:35.73
   55                                linker identification: "A11-49"
*****
File SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
   51                                image name: "LOC2"
   52                                image file identification: "V1.0"
   53                                image file build identification: ""
   54                                link date/time: 23-NOV-1998 15:37:49.68
   55                                linker identification: "A11-49"
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC1.ANA;1
   62   Analyze Image                                27-NOV-1998 23:36:16.68   P
   63   SYS$SYSROOT:[SYSMGR]LOC1.EXE;6
   64   ANALYZ A07-04
*****
File SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
   62   Analyze Image                                27-NOV-1998 23:36:23.32   P
   63   SYS$SYSROOT:[SYSMGR]LOC2.EXE;3
   64   ANALYZ A07-04
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC1.ANA;1
   122  Analyze Image                                27-NOV-1998 23:36:16.69   P
   123  SYS$SYSROOT:[SYSMGR]LOC1.EXE;6
   124  ANALYZ A07-04
*****
```

```

File SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
  122   Analyze Image                                27-NOV-1998 23:36:23.32   P
  123   SYS$SYSROOT:[SYSMGR]LOC2.EXE;3
  124   ANALYZ A07-04
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC1.ANA;1
  181   Analyze Image                                27-NOV-1998 23:36:16.70   P
  182   SYS$SYSROOT:[SYSMGR]LOC1.EXE;6
  183   ANALYZ A07-04
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
  181   Analyze Image                                27-NOV-1998 23:36:23.34   P
  182   SYS$SYSROOT:[SYSMGR]LOC2.EXE;3
  183   ANALYZ A07-04
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC1.ANA;1
  229   Analyze Image                                27-NOV-1998 23:36:16.70   P
  230   SYS$SYSROOT:[SYSMGR]LOC1.EXE;6
  231   ANALYZ A07-04
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
  229   Analyze Image                                27-NOV-1998 23:36:23.34   P
  230   SYS$SYSROOT:[SYSMGR]LOC2.EXE;3
  231   ANALYZ A07-04
*****
*****
File SYS$SYSROOT:[SYSMGR]LOC1.ANA;1
  284   ANA/IMAGE LOC1/OUT=LOC1.ANA
*****
File SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
  284   ANA/IMAGE LOC2/OUT=LOC2.ANA
*****
Number of difference sections found: 7
Number of difference records found: 15

DIFFERENCES /IGNORE=()/MERGED=1-
  SYS$SYSROOT:[SYSMGR]LOC1.ANA;1-
  SYS$SYSROOT:[SYSMGR]LOC2.ANA;1
$
$

```

```
$ set work/lim=1000/quo=1000/ext=1000
$ sh status
  Status on 27-NOV-1998 23:39:29.74      Elapsed CPU :   0 00:02:39.28
  Buff. I/O :      7434    Cur. ws. :   1008    Open files :         0
  Dir. I/O :      6266    Phys. Mem. :   1008    Page Faults :   30353
$ r loc1
$ sh status
  Status on 27-NOV-1998 23:39:35.49      Elapsed CPU :   0 00:02:41.00
  Buff. I/O :      7447    Cur. ws. :   1008    Open files :         0
  Dir. I/O :      6268    Phys. Mem. :    944    Page Faults :   32503
$
```

Note

LOC1 generates elapsed CPU time of 1.72 seconds, 2250 page faults.

```
$ show status
  Status on 27-NOV-1998 23:39:44.82      Elapsed CPU :   0 00:02:41.00
  Buff. I/O :      7450    Cur. ws. :   1008    Open files :         0
  Dir. I/O :      6268    Phys. Mem. :    944    Page Faults :   32503
$ r loc2
$ show status
  Status on 27-NOV-1998 23:43:37.31      Elapsed CPU :   0 00:06:21.92
  Buff. I/O :      7463    Cur. ws. :   1008    Open files :         0
  Dir. I/O :      6270    Phys. Mem. :    992    Page Faults :  4020293
$
```

Note

LOC2 generates elapsed CPU time of 3:40.92, 3,987,790 page faults

```
$ set work/lim=8000/quo=8000/ext=8000
%SET-I-NEWLIMS, new working set: Limit = 2848 Quota = 2848 Extent = 8000
$ show status
  Status on 27-NOV-1998 23:44:33.35      Elapsed CPU :   0 00:06:22.06
  Buff. I/O :      7495    Cur. ws. :   2848    Open files :         0
  Dir. I/O :      6270    Phys. Mem. :   1056    Page Faults :  4020418
$ r loc1
$ sh status
  Status on 27-NOV-1998 23:44:39.67      Elapsed CPU :   0 00:06:23.69
  Buff. I/O :      7508    Cur. ws. :   2848    Open files :         0
  Dir. I/O :      6272    Phys. Mem. :    992    Page Faults :  4022485
$
$ sh status
  Status on 27-NOV-1998 23:44:47.63      Elapsed CPU :   0 00:06:23.69
  Buff. I/O :      7513    Cur. ws. :   2848    Open files :         0
  Dir. I/O :      6272    Phys. Mem. :    992    Page Faults :  4022485
$ r loc2
$ sh status
  Status on 27-NOV-1998 23:48:33.28      Elapsed CPU :   0 00:09:57.60
  Buff. I/O :      7534    Cur. ws. :   2848    Open files :         0
  Dir. I/O :      6274    Phys. Mem. :    976    Page Faults :  7936902
$
$
```

```

$ set work/lim=16000/quo=16000/ext=16000
%SET-I-NEWLIMS, new working set: Limit = 2848 Quota = 2848 Extent = 16000
$ sh status
Status on 27-NOV-1998 23:49:30.31 Elapsed CPU : 0 00:09:57.76
Buff. I/O : 7569 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6274 Phys. Mem. : 1136 Page Faults : 7936951
$ r loc1
$ sh status
Status on 27-NOV-1998 23:49:36.29 Elapsed CPU : 0 00:09:59.42
Buff. I/O : 7586 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6276 Phys. Mem. : 992 Page Faults : 7939014
$ sh status
Status on 27-NOV-1998 23:49:43.43 Elapsed CPU : 0 00:09:59.42
Buff. I/O : 7590 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6276 Phys. Mem. : 992 Page Faults : 7939014
$ r loc2
$ sh status
Status on 27-NOV-1998 23:53:31.27 Elapsed CPU : 0 00:13:35.67
Buff. I/O : 7607 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6278 Phys. Mem. : 1008 Page Faults : 11849353
$

$ set work/lim=32000/quo=32000/ext=32000
%SET-I-NEWLIMS, new working set: Limit = 2848 Quota = 2848 Extent = 32000
$ show work
Working Set (pagelets) /Limit=2848 /Quota=2848 /Extent=32000
Adjustment enabled Authorized Quota=2848 Authorized Extent=73728

Working Set (8Kb pages) /Limit=178 /Quota=178 /Extent=2000
Authorized Quota=178 Authorized Extent=4608
$ show status
Status on 27-NOV-1998 23:54:25.52 Elapsed CPU : 0 00:13:35.95
Buff. I/O : 7648 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6278 Phys. Mem. : 1168 Page Faults : 11849446
$ r loc1
$ show status
Status on 27-NOV-1998 23:54:31.59 Elapsed CPU : 0 00:13:37.52
Buff. I/O : 7661 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6280 Phys. Mem. : 960 Page Faults : 11851509
$
$

```

\$ show status

Status on 27-NOV-1998 23:54:55.63 Elapsed CPU : 0 00:13:37.52
Buff. I/O : 7666 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6280 Phys. Mem. : 960 Page Faults : 11851509

\$ r loc2**\$ show status**

Status on 27-NOV-1998 23:55:04.17 Elapsed CPU : 0 00:13:43.63
Buff. I/O : 7683 Cur. ws. : 2848 Open files : 0
Dir. I/O : 6282 Phys. Mem. : 1024 Page Faults : 11882383

\$

Note

The numbers stay basically the same until the working set reaches 32000 pagelets (2000 Alpha pages). When loc2 takes only 6.11 seconds to run and generates 30874 page faults. This is 32 times the memory required to run loc1!

Although the programs look identical externally, loc1 has good locality (accessing in row order). The loc2 program has poor locality, accessing in column order.

```
$ type loc1.c
#include <stdlib.h>
#define NROWS 2000
#define NCOLS 2000
void stub(int);
int main(void)
{
    static int x[NROWS][NCOLS]={0};
    int i,j;

    for(i=0;i<NROWS;i++)
        for(j=0;j<NCOLS;j++)
        {
            x[i][j] += 42*i*j;
            stub(x[i][j]);
        }
    return(EXIT_SUCCESS);
}
$
$ type loc2.c
#include <stdlib.h>
#define NROWS 2000
#define NCOLS 2000
void stub(int);
int main(void)
{
    static int x[NROWS][NCOLS]={0};
    int i,j;

    for(j=0;j<NCOLS;j++)
        for(i=0;i<NROWS;i++)
        {
            x[i][j] += 42*i*j;
            stub(x[i][j]);
        }
    return(EXIT_SUCCESS);
}
}
```

The subroutine stub and the method of call blocks the optimizer from dropping the code.

```
$ type stub.c
void stub(int x)
{
}
$
```

Module 7 Labs (Shared Image Costs and Benefits)

1. Copy the files PERF\$:big_code.exe and big_code.com and big_code_100 to your directory.
 - a. Issue the command \$ @big_code. Determine the total number of faults generated on the RUN of the program and the elapsed time for execution.

```
$ @big_code
starting execution
  Status on 28-NOV-1998 00:24:26.69      Elapsed CPU : 0 00:00:30.07
  Buff. I/O :      2838      Cur. ws. : 1424      Open files :      1
  Dir. I/O :      9421      Phys. Mem. : 1232      Page Faults :     352
  Status on 28-NOV-1998 00:24:26.76      Elapsed CPU : 0 00:00:30.13
  Buff. I/O :      2843      Cur. ws. : 3824      Open files :      1
  Dir. I/O :      9424      Phys. Mem. : 1280      Page Faults :     363
ending execution
$
```

The elapsed time of execution is .07 seconds. The elapsed CPU time is .06 seconds. The number of faults is 11.

- b. INSTALL BIG_CODE.EXE with the /SHARE qualifier. Repeat the method in step "a" twice. Compare the numbers. Explain the difference.

```
$ INSTALL ADD SYS$LOGIN:BIG_CODE/SHARE
$ @big_code
starting execution
  Status on 28-NOV-1998 00:25:33.33      Elapsed CPU : 0 00:00:30.77
  Buff. I/O :      2925      Cur. ws. : 1424      Open files :      1
  Dir. I/O :      9438      Phys. Mem. : 1168      Page Faults :     696
  Status on 28-NOV-1998 00:25:33.46      Elapsed CPU : 0 00:00:30.82
  Buff. I/O :      2928      Cur. ws. : 1424      Open files :      1
  Dir. I/O :      9440      Phys. Mem. : 1104      Page Faults :     725
ending execution
```

On the first run, after installation, the elapsed time is .13 seconds, elapsed CPU time is .05 seconds, and page faults are 29.

```
$ @big_code
starting execution
  Status on 28-NOV-1998 00:25:35.25      Elapsed CPU : 0 00:00:30.84
  Buff. I/O :      2937      Cur. ws. : 3824      Open files :      1
  Dir. I/O :      9441      Phys. Mem. : 1184      Page Faults :     730
  Status on 28-NOV-1998 00:25:35.30      Elapsed CPU : 0 00:00:30.86
  Buff. I/O :      2940      Cur. ws. : 1424      Open files :      1
  Dir. I/O :      9443      Phys. Mem. : 1184      Page Faults :     774
ending execution
$
```

On the second run, after installation, the elapsed time is .05 seconds, elapsed CPU time is .02 seconds, and page faults are 44.



Note

The reason for the drop in elapsed time and elapsed CPU time, is due to the I/Os that were required in the first run to get the image into memory. The increase in page faults is due to more soft faults (which do not prefetch, as hard faults do).

- c. Determine whether BIG_CODE.EXE can be installed with the /RESIDENT qualifier. If not copy the object module BIG_CODE.OBJ and link it so that it can be installed /RESIDENT. Install it /RESIDENT and compare the numbers.

```
$ more="type/page=save sys$input"
$
$ pipe ana/image big_code | more
Analyze Image                                28-NOV-1998 00:27:43.54   Page
1
SYS$SYSROOT:[SYSMGR]BIG_CODE.EXE;2
ANALYZ A07-04
```

This is an OpenVMS Alpha image file

IMAGE HEADER

Fixed Header Information

image format major id: 3, minor id: 0

...

linker flags:

...

```
(8) EIHD$V_BIND_CODE 0
      (9) EIHD$V_BIND_DATA 0
      (10) EIHD$V_MKTHREADS 0
      (11) EIHD$V_UPCALLS 0
      (12) EIHD$V_OMV_READY 0
      (13) EIHD$V_EXT_BIND_ 0
```

...

The image cannot be installed resident, as it is, since the flags EIHD\$V_BIND_CODE and EIHD\$V_BIND_DATA are clear.

```
$ link/section=(code,data)/NOTRACE big_code
$
```

Installing /RESIDENT and /SHARE=ADDRESS reduce the Elapsed time to .04 seconds and the Page faults to 5.

```
$ install add sys$login:big_code/resident/share=address
$ @big_code
starting execution
Status on 28-NOV-1998 00:32:30.23      Elapsed CPU : 0 00:00:36.16
Buff. I/O :      3568      Cur. ws. : 1424      Open files :      1
Dir. I/O :      10436      Phys. Mem. : 1232      Page Faults :      2740
Status on 28-NOV-1998 00:32:30.27      Elapsed CPU : 0 00:00:36.18
Buff. I/O :      3570      Cur. ws. : 1424      Open files :      1
Dir. I/O :      10436      Phys. Mem. : 1264      Page Faults :      2745
ending execution
$
```

d. Run the command procedure BIG_CODE_100.COM with:

- ♦ The image uninstalled.
- ♦ The image installed /SHARE.
- ♦ The image installed /RESIDENT.
- ♦ /RESIDENT and /SHARE=ADDRESS_DATA

Compare execution times. Examine the paging statistics from \$MONITOR PAGE, while the command procedures are running.

```
$ install delete sys$login:big_code
$ @big_code_100
starting execution
Status on 28-NOV-1998 00:34:43.45      Elapsed CPU : 0 00:00:36.41
Buff. I/O :      3609      Cur. ws. : 1424      Open files :      1
Dir. I/O :      10441      Phys. Mem. : 1296      Page Faults :      2813
Status on 28-NOV-1998 00:34:48.60      Elapsed CPU : 0 00:00:41.17
Buff. I/O :      4010      Cur. ws. : 1424      Open files :      1
Dir. I/O :      10541      Phys. Mem. : 1072      Page Faults :      4321
ending execution
```

Elapsed time == 5.15 seconds

```
$ install add      sys$login:big_code/share
$ @big_code_100
starting execution
Status on 28-NOV-1998 00:35:06.58      Elapsed CPU : 0 00:00:41.44
Buff. I/O :      4049      Cur. ws. : 3824      Open files :      1
Dir. I/O :      10547      Phys. Mem. : 1376      Page Faults :      4424
Status on 28-NOV-1998 00:35:09.03      Elapsed CPU : 0 00:00:43.71
Buff. I/O :      4250      Cur. ws. : 1424      Open files :      1
Dir. I/O :      10647      Phys. Mem. : 1040      Page Faults :      7237
ending execution
$
```

Elapsed time == 2.45 seconds

```

$ install replace sys$login:big_code/resident
$ @big_code_100
starting execution
Status on 28-NOV-1998 00:35:48.48      Elapsed CPU : 0 00:00:43.97
Buff. I/O : 4288      Cur. ws. : 1424      Open files : 1
Dir. I/O : 10654      Phys. Mem. : 1360      Page Faults : 7347
Status on 28-NOV-1998 00:35:50.52      Elapsed CPU : 0 00:00:45.88
Buff. I/O : 4489      Cur. ws. : 1424      Open files : 1
Dir. I/O : 10654      Phys. Mem. : 1360      Page Faults : 7671
ending execution
$
    Elapsed time == 2.04 seconds

```

```

$ install delete sys$login:big_code
$ install add sys$login:big_code/resident/share=address
$ @big_code_100
starting execution
Status on 28-NOV-1998 00:36:39.39      Elapsed CPU : 0 00:00:46.65
Buff. I/O : 4609      Cur. ws. : 3824      Open files : 1
Dir. I/O : 10674      Phys. Mem. : 1360      Page Faults : 7932
Status on 28-NOV-1998 00:36:41.24      Elapsed CPU : 0 00:00:48.39
Buff. I/O : 4710      Cur. ws. : 1424      Open files : 1
Dir. I/O : 10674      Phys. Mem. : 1040      Page Faults : 8285
ending execution
$
    Elapsed time == 1.45 seconds

```

MONITOR PAGE behavior with:

- ♦ The image un-installed.
High Page Read I/O Rate and Page Read Rate
- ♦ The image installed /SHARE.
Still some Page Read I/Os, higher Free List and Global Valid Rates.
- ♦ The image installed /RESIDENT
Still some Page Read I/Os.
- ♦ /RESIDENT and /SHARE=ADDRESS_DATA
Page Read I/Os go away.

2. Determine whether the image SYSS\$SYSTEM:SDA.EXE may be installed /RESIDENT.

It may not, since the BIND_CODE and BIND_DATA flags are clear.

```
$ ANALYZE/IMAGE SYSS$SYSTEM:SDA
```

```
Analyze Image
```

```
28-NOV-1998 00:41:15.85 Page
```

```
1
```

```
SYSS$COMMON:[SYSEXEX]SDA.EXE;1 ANALYZ A07-04
```

This is an OpenVMS Alpha image file

```
IMAGE HEADER
```

```
...
```

```
linker flags:
```

```
(8) EIHD$V_BIND_CODE 0
```

```
(9) EIHD$V_BIND_DATA 0
```

```
...
```

3. Install loc2 /SHARE and compare the paging statistics/execution time to when it is not installed. Do they change?

The numbers do not change significantly for loc2, since the faults are caused by accessing data.

```
$ inst add sys$login:loc2/share
$ sh status
Status on 28-NOV-1998 00:42:40.83      Elapsed CPU :   0 00:01:05.78
Buff. I/O :      6491    Cur. ws. :   1424    Open files :      0
Dir. I/O :      10942    Phys. Mem. :   1136    Page Faults :   15454
$ r loc2
$ show status
$ show status
Status on 28-NOV-1998 00:42:50.47      Elapsed CPU :   0 00:01:12.41
Buff. I/O :      6503    Cur. ws. :   1424    Open files :      0
Dir. I/O :      10944    Phys. Mem. :    992    Page Faults :   49965
$ set work/lim=16000/quo=16000/ext=16000
%SET-I-NEWLIMS, new working set: Limit = 2848 Quota = 2848 Extent = 16000
$ show status
Status on 28-NOV-1998 00:43:24.27      Elapsed CPU :   0 00:01:12.53
Buff. I/O :      6520    Cur. ws. :   2848    Open files :      0
Dir. I/O :      10944    Phys. Mem. :   1008    Page Faults :   50059
$ r loc2
$ show status
Status on 28-NOV-1998 00:47:32.75      Elapsed CPU :   0 00:05:02.36
Buff. I/O :      6543    Cur. ws. :   2848    Open files :      0
Dir. I/O :      10946    Phys. Mem. :    976    Page Faults :  3960399
$
```

Module 8 Labs (Modified Page List and Modified Page Writing)

- Determine the number of pages being written per Modified Page Write.

```

OpenVMS Monitor Utility
PAGE MANAGEMENT STATISTICS
  on node ELLIS
28-NOV-1998 00:45:08.48

CUR      AVE      MIN      MAX

...
Page Write Rate      0.00      0.00      0.00      0.00
Page Write I/O Rate  0.00      0.00      0.00      0.00
...
PAGE MANAGEMENT STATISTICS
      on node ELLIS
28-NOV-1998 00:52:37.75

CUR      AVE      MIN      MAX

...
Page Write Rate      170.66      5.07      0.00      170.66
Page Write I/O Rate   2.66      0.07      0.00      2.66
...

```

The number of pages being written per I/O is equal to the "Write rate" divided by the "Write I/O rate". In the first case no pages are being written. In the second case 64 pages are being written per I/O, which should approximate the SYSGEN parameter MPW_WRTCLUSTER.

- Determine the amount of space in use in the page file(s).

In this case, none.

```

$ show mem/file/full
      System Memory Resources on 28-NOV-1998 00:45:40.42

DISK$AXP71FT2:[SYS0.SYSEXE]SWAPFILE.SYS
  Free Blocks      20480      Reservable Blocks      20480
  Total Size (blocks) 20480      Paging File Number      1
  Swap Usage (processes) 0      Paging Usage (processes) 0
  This file is used exclusively for swapping.

DISK$AXP71FT2:[SYS0.SYSEXE]PAGEFILE.SYS
  Free Blocks      290816      Reservable Blocks      151216
  Total Size (blocks) 290816      Paging File Number      3
  Swap Usage (processes) 0      Paging Usage (processes) 30
  This file can be used for either paging or swapping.
$

```

Module 9 Labs (The Swapper and Memory Reclamation)

1. Determine whether any processes on the system are swapped.

In this case, none are swapped.

```
$ sh mem/slot
System Memory Resources on 28-NOV-1998 00:54:25.54

Slot Usage (slots):          Total      Free      Resident      Swapped
Process Entry Slots          136       103         33           0
Balance Set Slots            134       103         31           0
$
```

2. Set FREEGOAL to the size of the free page list plus 1000. Determine whether any processes are swapped-out.

```
$ sh mem/phy
System Memory Resources on 28-NOV-1998 00:55:05.99

Physical Memory Usage (pages):  Total      Free      In Use      Modified
Main Memory (160.00Mb)         20480     11503       8805        172

Of the physical pages in use, 3583 pages are permanently allocated to OpenVMS.
$ mcr sysgen
SYSGEN> SET FREEG 12500
SYSGEN> WRITE ACT
SYSGEN> Exit
$
```



Note

Swapping occurs when the Free List size drops below FREEGOAL pages in size.

\$ sh mem/slot

System Memory Resources on 28-NOV-1998 00:55:35.90

Slot Usage (slots):	Total	Free	Resident	Swapped
Process Entry Slots	136	103	31	2
Balance Set Slots	134	103	29	2

\$

\$ sh sys

Pid	Process Name	State	Pri	I/O	CPU	Page flts	Pages
OpenVMS K7.2	on node ELLIS	28-NOV-1998	00:56:05.38	Uptime	0 02:52:06		

Pid	Process Name	State	Pri	I/O	CPU	Page flts	Pages
20200101	SWAPPER	HIB	16	0	0 00:00:00.57	0	0
20200106	CLUSTER_SERVER	HIB	13	13	0 00:00:00.14	60	77
20200107	CONFIGURE	HIB	10	12	0 00:00:00.28	43	14
20200108	LANACP	HIBO	13	--	swapped out --		29
2020010A	FASTPATH_SERVER	HIB	12	9	0 00:00:00.16	73	86
2020010B	IPCACP	HIB	10	10	0 00:00:00.07	36	27
2020010C	ERRFMT	HIB	9	326	0 00:00:01.45	91	79
2020010D	CACHE_SERVER	HIBO	16	--	swapped out --		32
2020010E	OPCOM	HIB	8	111	0 00:00:00.36	76	25
2020010F	AUDIT_SERVER	HIB	10	59	0 00:00:00.37	142	129
20200110	JOB_CONTROL	HIBO	10	--	swapped out --		32
20200111	QUEUE_MANAGER	HIBO	9	--	swapped out --		32
20200112	SECURITY_SERVER	HIB	10	32	0 00:01:01.98	492	176
20200113	DNS\$ADVER	LEFO	5	--	swapped out --		26
20200114	LES\$ACP_V30	HIBO	8	--	swapped out --		30
20200115	REMACP	HIBO	8	--	swapped out --		16
20200116	NET\$ACP	HIB	4	45	0 00:00:02.95	137	111
20200117	NET\$EVD	HIB	6	20	0 00:00:00.86	251	249
20200118	DTSS\$CLERK	LEF	11	174	0 00:00:01.00	139	142
20200119	SMISERVER	HIB	9	42	0 00:00:00.54	176	134

...

\$ sh mem/slot

System Memory Resources on 28-NOV-1998 00:56:36.64

Slot Usage (slots):	Total	Free	Resident	Swapped
Process Entry Slots	136	103	24	9
Balance Set Slots	134	103	22	9

\$

\$ sh mem/slot

System Memory Resources on 28-NOV-1998 00:56:59.10

Slot Usage (slots):	Total	Free	Resident	Swapped
Process Entry Slots	136	103	21	12
Balance Set Slots	134	103	19	12

\$

\$ sh mem/slot

System Memory Resources on 28-NOV-1998 00:57:16.24

Slot Usage (slots):	Total	Free	Resident	Swapped
Process Entry Slots	136	103	18	15
Balance Set Slots	134	103	16	15

\$

3. At what point will the system perform proactive memory reclamation?

Based on the following information, "tickling" will occur when the Free List has less than 1072 pages on it. "Trolling" will occur when the Free List has less than 536 pages on it.

```
$ mcr sysgen show mmg_ctlflags
Parameter Name      Current  Default  Min.    Max.    Unit Dynamic
-----
MMG_CTLFLAGS        3        3        0       -1    Bitmask   D
$ mcr sysgen show freegoal
Parameter Name      Current  Default  Min.    Max.    Unit Dynamic
-----
FREEGOAL            536     200     16       -1    Pages     D
$
$
```

Module 10 Labs (System Faults)

1. Determine the system fault rate.

In this case, the System Fault Rate is 0.

OpenVMS Monitor Utility
PAGE MANAGEMENT STATISTICS
on node ELLIS
28-NOV-1998 00:58:35.13

	CUR	AVE	MIN	MAX
Page Fault Rate	0.00	7449.70	0.00	17459.33
Page Read Rate	0.00	0.82	0.00	19.66
Page Read I/O Rate	0.00	0.45	0.00	9.66
Page Write Rate	0.00	2.09	0.00	170.66
Page Write I/O Rate	0.00	0.03	0.00	2.66
Free List Fault Rate	0.00	3.75	0.00	22.66
Modified List Fault Rate	0.00	7379.25	0.00	17236.66
Demand Zero Fault Rate	0.00	14.22	0.00	588.66
Global Valid Fault Rate	0.00	51.92	0.00	225.33
Wrt In Progress Fault Rate	0.00	0.00	0.00	0.00
System Fault Rate	0.00	0.00	0.00	0.00
Free List Size	12302.00	9529.57	5564.00	12339.00
Modified List Size	1561.00	2112.64	171.00	4583.00

2. Determine the size of the system working set list.

\$ mcr sysgen show sysmwcnt

Parameter Name	Current	Default	Min.	Max.	Unit Dynamic
-----	-----	-----	-----	-----	-----
SYSMWcnt	4362	2048	512	1048576	Pagelets
internal value	273	128	32	65536	Pages

\$

3. Issue the command `$@PERF$:touch_ppool`. Determine the system fault rate while it is running.

From one window/terminal:

```
$ @touch_ppool
```

```
OpenVMS (TM) Alpha system analyzer
```

```
$
```

From another:

```

                                OpenVMS Monitor Utility
                                PAGE MANAGEMENT STATISTICS
                                on node ELLIS
                                28-NOV-1998 01:02:11.34

```

	CUR	AVE	MIN	MAX
Page Fault Rate	15.00	5501.36	0.00	17459.33
Page Read Rate	0.00	1.21	0.00	39.00
Page Read I/O Rate	0.00	0.54	0.00	14.33
Page Write Rate	0.00	1.54	0.00	170.66
Page Write I/O Rate	0.00	0.02	0.00	2.66
Free List Fault Rate	7.00	3.13	0.00	22.66
Modified List Fault Rate	8.00	5447.66	0.00	17236.66
Demand Zero Fault Rate	0.00	11.19	0.00	588.66
Global Valid Fault Rate	0.00	38.75	0.00	225.33
Wrt In Progress Fault Rate	0.00	0.00	0.00	0.00
System Fault Rate	15.00	0.24	0.00	52.66
Free List Size	12182.00	10225.01	5564.00	12339.00
Modified List Size	1638.00	1968.64	171.00	4583.00

Module 11 Labs (General I/O Flow)

1. Determine the Virtual I/O cache hit rate on the system.

```
$ show memory/cache/full
      System Memory Resources on 28-NOV-1998 01:03:07.42

Virtual I/O Cache
  Total Size (Kbytes)      3200    Read IO Count      30053
  Free Kbytes              8       Read Hit Count     20738
  Kbytes in Use            3192    Read Hit Rate      69%
  Write IO Bypassing Cache 0       Write IO Count     961
  Files Retained           97     Read IO Bypassing Cache 1457
$
```

2. Determine the I/O rates and queue depths for each disk.

```
$ monitor disk/item=all
      OpenVMS Monitor Utility
      DISK I/O STATISTICS
      on node ELLIS
      28-NOV-1998 01:08:33.21

I/O Operation Rate          CUR      AVE      MIN      MAX

ELLIS$DKA200:      AXP71FT2      0.00      0.12      0.00      1.00
```

```
      OpenVMS Monitor Utility
      DISK I/O STATISTICS
      on node ELLIS
      28-NOV-1998 01:08:06.18

I/O Request Queue Length    CUR      AVE      MIN      MAX

ELLIS$DKA200:      AXP71FT2      0.00      0.00      0.00      0.00
```

Module 12 Labs (Analyzing the Effects of Fragmentation)

1. Run the program `PERF$:fragger` (use `$SHOW STATUS` to determine the Elapsed time of execution and I/Os generated by the program). Determine the:
 - a. Split transfer rate.
 - b. Window turn rate.
 - c. I/O rates to all disks.
 - d. the top direct I/O processes.

Sample run of fragger generates 7713 Direct I/Os and 3:20.40 elapsed time.

```
$ sh status
Status on 28-NOV-1998 12:48:39.59      Elapsed CPU : 0 00:01:13.48
Buff. I/O : 3291      Cur. ws. : 1424      Open files : 0
Dir. I/O : 11126      Phys. Mem. : 1200      Page Faults : 5631
$ r fragger
$ sh status
Status on 28-NOV-1998 12:51:59.99      Elapsed CPU : 0 00:01:38.36
Buff. I/O : 4838      Cur. ws. : 1424      Open files : 0
Dir. I/O : 18839      Phys. Mem. : 1200      Page Faults : 5675
$
```

On another terminal/window we monitor the performance.

```
$ monitor/interval=10 io, fcp, disk, process/topdio
```

```
OpenVMS Monitor Utility
I/O SYSTEM STATISTICS
on node ELLIS
28-NOV-1998 12:45:16.34
```

	CUR	AVE	MIN	MAX
Direct I/O Rate	41.79	15.96	2.79	41.79
Buffered I/O Rate	9.80	3.82	0.60	9.80
Mailbox Write Rate	0.00	0.00	0.00	0.00
Split Transfer Rate	9.00	2.78	0.00	9.00
Log Name Translation Rate	0.00	2.86	0.00	10.77
File Open Rate	0.00	0.09	0.00	0.29
Page Fault Rate	0.20	1.64	0.20	4.99
Page Read Rate	0.00	0.23	0.00	1.39
Page Read I/O Rate	0.00	0.14	0.00	0.89
Page Write Rate	0.00	0.00	0.00	0.00
Page Write I/O Rate	0.00	0.00	0.00	0.00
Inswap Rate	0.00	0.00	0.00	0.00
Free List Size	10625.00	10648.00	10625.00	10671.00
Modified List Size	1158.00	1158.33	1154.00	1161.00

Note

A split transfer rate of 9.00 per second, indicating fragmentation. There is also a high direct I/O rate of 41.79 per second.

OpenVMS Monitor Utility
 FILE PRIMITIVE STATISTICS
 on node ELLIS
 28-NOV-1998 12:45:26.35

	CUR	AVE	MIN	MAX
FCP Call Rate	9.69	3.99	0.10	9.69
Allocation Rate	8.19	3.45	0.00	9.50
Create Rate	0.00	0.05	0.00	0.39
Disk Read Rate	0.00	0.07	0.00	0.29
Disk Write Rate	16.29	5.06	0.00	16.29
Volume Lock Wait Rate	0.00	0.00	0.00	0.00
CPU Tick Rate	5.40	1.91	0.00	5.40
File Sys Page Fault Rate	0.00	0.00	0.00	0.00
Window Turn Rate	8.00	3.39	0.00	9.60
File Lookup Rate	1.10	0.37	0.00	1.10
File Open Rate	0.30	0.12	0.00	0.30
Erase Rate	8.19	3.72	0.00	9.39

Note

A window turn rate of 8.00 per second, also indicating fragmentation.

OpenVMS Monitor Utility
 DISK I/O STATISTICS
 on node ELLIS
 28-NOV-1998 12:45:36.36

I/O Operation Rate		CUR	AVE	MIN	MAX
ELLIS\$DKA200:	AXP71FT2	48.70	22.99	0.00	50.09

The device dka200: is absorbing 48.7 I/Os per second. Note: this is approximately the same as the sum of the direct I/O rate and the split transfer rate.

OpenVMS Monitor Utility
 TOP DIRECT I/O RATE PROCESSES
 on node ELLIS
 28-NOV-1998 12:47:06.43

	0	5	10	15	20
	+ - - - -	+ - - - -	+ - - - -	+ - - - -	+ - - - -
20200128 _FTA2:	40				
2020011E DECW\$SERVER_0	2				
20200123 DTWM					
	+ - - - -	+ - - - -	+ - - - -	+ - - - -	+ - - - -

The process "_FTA2:" is generating the majority of the direct I/Os. Let us see which files the process is accessing.

```
$ pipe show dev/files dka200 | search sys$input _FTA2:
_FTA2:      20200128 [SYS0.SYSMGR]FRAGGER.EXE;2
_FTA2:      20200128 [SYS0.SYSMGR]X.DAT;2
_FTA2:      20200128 [SYS0.SYSMGR]Y.DAT;2
$
```

SSHOW DEVICE/FILES shows that X.dat and Y.dat are being accessed. It would appear that they are fragmented.

Determine how fragmented the files are by examining the file header and looking at the mapping pointers for the files.

```
$ more="type/page=save sys$input"
$ pipe dump/header/block=c:0 dka200:[sys0.sysmgr]x.dat | more
Dump of file DKA200:[SYS0.SYSMGR]X.DAT;2 on 28-NOV-1998 12:55:08.25
File ID (6833,4,0) End of file block 25000 / Allocated 25002
```

File Header

Header area

```
Identification area offset:      40
Map area offset:                 100
Access control area offset:      255
Reserved area offset:            255
Extension segment number:        0
Structure level and version:     2, 1
File identification:              (6833,4,0)
```

Note

An extension file id may indicate that there are so many mapping pointers that we cannot fit them in a single header block.

```

Extension file identification:      (6836,4,0)
VAX-11 RMS attributes
  Record type:                     Fixed
  File organization:               Sequential
  Record attributes:               <none specified>
  Record size:                     512
  Highest block:                   25002
  End of file block:               25001
  End of file byte:                0
  End of file byte:                0
  Bucket size:                    0
  Fixed control area size:         0
  Maximum record size:             512
  Default extension size:          0
  Global buffer count:             0
  Directory version limit:         0
File characteristics:              <none specified>
Caching attribute:                 Writethrough
Map area words in use:             154
Access mode:                       0
File owner UIC:                    [SYSTEM]
File protection:                   S:RWED, O:RWED, G:RE, W:
Back link file identification:      (2396,1,0)
Journal control flags:             <none specified>
Active recovery units:             None
Highest block written:             25002
Client attributes:                 None

Identification area
  File name:                       X.DAT;2
  Revision number:                  1
  Creation date:                   28-NOV-1998 12:48:42.75
  Creation date:                   28-NOV-1998 12:48:42.75
  Revision date:                   28-NOV-1998 12:51:59.83
  Expiration date:                 <none specified>
  Backup date:                     <none specified>

```

The start of the mapping/Retrieval pointers. Each mapping pointer represents a distinct extent. The file appears to be very fragmented.

Map area

Retrieval pointers

Count:	3	LBN:	135714
Count:	9	LBN:	135744
Count:	3	LBN:	136539
Count:	9	LBN:	137316
Count:	9	LBN:	137691
Count:	3	LBN:	137703
Count:	27	LBN:	159654
Count:	6	LBN:	159714
Count:	27	LBN:	171591
Count:	6	LBN:	171651

Dump of file DKA200:[SYS0.SYSMGR]X.DAT;2 on 28-NOV-1998 12:55:08.25

File ID (6833,4,0) End of file block 25000 / Allocated 25002

Count:	27	LBN:	171723
Count:	6	LBN:	171783
Count:	27	LBN:	171855
Count:	6	LBN:	171915
Count:	27	LBN:	172023
Count:	6	LBN:	177486
Count:	27	LBN:	177558
Count:	6	LBN:	177618
Count:	27	LBN:	177690
Count:	6	LBN:	177750
Count:	27	LBN:	177822
Count:	6	LBN:	177882
Count:	33	LBN:	559665
Count:	33	LBN:	559731
Count:	33	LBN:	559797
Count:	33	LBN:	559863
Count:	33	LBN:	559929
Count:	33	LBN:	559995
Count:	33	LBN:	560061
Count:	21	LBN:	603342
Count:	12	LBN:	603414
Count:	15	LBN:	626649
Count:	18	LBN:	628413
Count:	15	LBN:	628497
Count:	18	LBN:	628545
Count:	15	LBN:	628629
Count:	18	LBN:	628677
Count:	15	LBN:	628761
Count:	18	LBN:	628809
Count:	15	LBN:	628893
Count:	18	LBN:	628941
Count:	15	LBN:	629025
Count:	18	LBN:	629073
Count:	33	LBN:	626688
Count:	33	LBN:	626754

Count:	33	LBN:	626820
Count:	33	LBN:	626886
Count:	33	LBN:	626952
Count:	33	LBN:	627018
Count:	33	LBN:	627084
Count:	33	LBN:	627150
Count:	33	LBN:	627216
Count:	33	LBN:	627282
Count:	33	LBN:	627348
Count:	33	LBN:	627414
Count:	33	LBN:	627480
Count:	33	LBN:	627546
Count:	33	LBN:	627612
Count:	33	LBN:	627678
Count:	33	LBN:	627744
Count:	33	LBN:	627810
Count:	33	LBN:	627876
Count:	33	LBN:	627942
Count:	33	LBN:	628008
Count:	33	LBN:	628074
Count:	33	LBN:	628140
Count:	33	LBN:	628206

Dump of file DKA200:[SYS0.SYSMGR]X.DAT;2 on 28-NOV-1998 12:55:08.25
 File ID (6833,4,0) End of file block 25000 / Allocated 25002

Count:	33	LBN:	628272
Count:	33	LBN:	628338
Count:	9	LBN:	628404
Count:	24	LBN:	628446
Count:	9	LBN:	628536
Count:	24	LBN:	628578
Count:	9	LBN:	628668
Count:	9	LBN:	628668
Count:	24	LBN:	628710
Count:	9	LBN:	628800
Count:	24	LBN:	628842

Checksum: 2100

The extension header shows more mapping pointers and an additional extension header.

Dump of file DKA200:[SYS0.SYSMGR]X.DAT;2 on 28-NOV-1998 12:55:08.25
 File ID (6833,4,0) End of file block 25000 / Allocated 25002

File Header

Header area

Identification area offset:	40
Map area offset:	50
Access control area offset:	255
Reserved area offset:	255
Extension segment number:	1
Structure level and version:	2, 1
File identification:	(6836,4,0)
Extension file identification:	(6838,4,0)
VAX-11 RMS attributes	
Record type:	Fixed

...

Map area

Retrieval pointers

Count:	9	LBN:	628932
Count:	24	LBN:	628974
Count:	9	LBN:	629064
Count:	24	LBN:	629106
Count:	33	LBN:	629181
Count:	33	LBN:	629247
Count:	33	LBN:	629313
Count:	33	LBN:	629379
Count:	33	LBN:	629445
Count:	33	LBN:	629511
Count:	33	LBN:	629577
Count:	33	LBN:	629643
Count:	33	LBN:	629709
Count:	33	LBN:	629775
Count:	33	LBN:	629841

Dump of file DKA200:[SYS0.SYSMGR]X.DAT;2 on 28-NOV-1998 12:55:08.25
 File ID (6833,4,0) End of file block 25000 / Allocated 25002

Count:	33	LBN:	629907
Count:	33	LBN:	629973
Count:	33	LBN:	630039
Count:	33	LBN:	630105
Count:	33	LBN:	630171
Count:	33	LBN:	630237
Count:	33	LBN:	630303
Count:	33	LBN:	630369
Count:	33	LBN:	630435
Count:	33	LBN:	630501
Count:	33	LBN:	630567
Count:	33	LBN:	630633
Count:	33	LBN:	630699
Count:	12	LBN:	630765
Count:	21	LBN:	640194
Count:	33	LBN:	640248
Count:	33	LBN:	640314
Count:	33	LBN:	640380

...

Count the number of mapping pointers for each file.

Note

X.DAT is made up of 803 extents, while Y.DAT is made up of 802.

\$ pipe dump/header/block=c:0 x.dat | search/stat/nooutput sys\$input "LBN:"

Files searched:	1	Buffered I/O count:	1306
Records searched:	1304	Direct I/O count:	0
Characters searched:	57834	Page faults:	26
Records matched:	803	Elapsed CPU time:	0 00:00:00.76
Lines printed:	0	Elapsed time:	0 00:00:02.30

\$ pipe dump/header/block=c:0 y.dat | search/stat/nooutput sys\$input "LBN:"

Files searched:	1	Buffered I/O count:	1305
Records searched:	1303	Direct I/O count:	0
Characters searched:	57787	Page faults:	26
Records matched:	802	Elapsed CPU time:	0 00:00:00.93
Lines printed:	0	Elapsed time:	0 00:00:03.33

\$

- Determine which files fragger creates. Determine the level of fragmentation in the files. Without changing the program can you get it to create files that are more contiguous.

Although you cannot guarantee contiguity, by preallocating space for the files, we may be able to minimize the fragmentation. The following display shows that the default extend size for the process is 0, which implies that RMS will calculate the extend size. The calculated extend size is dependent on the cluster factor for the disk, however, it is typically around 33 blocks. This size is reflected by the largest extent sizes in the previous example.

```
$ SHOW RMS
```

	MULTI- BLOCK COUNT		Indexed	Relative	MULTIBUFFER COUNTS					NETWORK BLOCK COUNT
					Sequential					
					Disk	Magtape	Unit	Record		
Process	0		0	0	0	0	0	0		0
System	16		0	0	0	0	0	0		8

	Prolog	Extend	Quantity	VCC_DFW
Process	0		0	0
System	0		0	0

```
$
```

Determine the size of the files.

```
$ dir/siz x.dat
```

```
Directory SYS$SYSROOT:[SYSMGR]
```

```
X.DAT;2          25000
X.DAT;1          832
```

```
Total of 2 files, 25832 blocks.
```

Try to grab enough space on the first extent of the file to hold the entire file.

```
$ SET RMS/EXT=25000
$ sh status
Status on 28-NOV-1998 13:03:27.80 Elapsed CPU : 0 00:01:42.61
Buff. I/O : 5494 Cur. ws. : 3824 Open files : 0
Dir. I/O : 18977 Phys. Mem. : 1200 Page Faults : 6409
$ r fragger
$ sh status
Status on 28-NOV-1998 13:04:53.98 Elapsed CPU : 0 00:01:52.25
Buff. I/O : 5515 Cur. ws. : 1424 Open files : 0
Dir. I/O : 22179 Phys. Mem. : 1264 Page Faults : 6466
$
```

Note

The previous run took 1:26.18 of elapsed time and generated 4202 Direct I/Os. This is over twice as fast as the previous run. The improvement in performance cannot be guaranteed and could indeed be better.

The new files are described by 30 and 19 extents. This is a significant reduction in fragmentation.

```
$ pipe dump/header/block=c:0 x.dat | search/stat/nooutput sys$input "LBN:"
Files searched: 1 Buffered I/O count: 89
Records searched: 87 Direct I/O count: 0
Characters searched: 3742 Page faults: 26
Records matched: 30 Elapsed CPU time: 0 00:00:00.07
Lines printed: 0 Elapsed time: 0 00:00:00.37
$ pipe dump/header/block=c:0 y.dat | search/stat/nooutput sys$input "LBN:"
Files searched: 1 Buffered I/O count: 78
Records searched: 76 Direct I/O count: 0
Characters searched: 3214 Page faults: 26
Records matched: 19 Elapsed CPU time: 0 00:00:00.06
Lines printed: 0 Elapsed time: 0 00:00:00.19
$
```

The file header shows no extension headers.

```
$ pipe dump/header/block=c:0 dka200:[sys0.sysmgr]x.dat | more
Dump of file DKA200:[SYS0.SYSMGR]X.DAT;3 on 28-NOV-1998 13:06:09.94
File ID (6820,4,0)   End of file block 25000 / Allocated 25002
```

File Header

Header area

Identification area offset:	40
Map area offset:	100
Access control area offset:	255
Reserved area offset:	255
Extension segment number:	0
Structure level and version:	2, 1
File identification:	(6820,4,0)
Extension file identification:	(0,0,0)
VAX-11 RMS attributes	
Record type:	Fixed
File organization:	Sequential
Record attributes:	<none specified>
Record size:	512
Highest block:	25002
End of file block:	25001

...

Map area

Retrieval pointers

Count:	3	LBN:	148161
Count:	21	LBN:	148167
Count:	864	LBN:	148191
Count:	114	LBN:	149460
Count:	450	LBN:	742671
Count:	6	LBN:	743511
Count:	1434	LBN:	743520
Count:	75	LBN:	746397
Count:	12	LBN:	748314
Count:	1002	LBN:	749577

```
Dump of file DKA200:[SYS0.SYSMGR]X.DAT;3 on 28-NOV-1998 13:06:09.94
File ID (6820,4,0)   End of file block 25000 / Allocated 25002
```

Count:	3	LBN:	750654
Count:	6	LBN:	753240
Count:	3	LBN:	750654
Count:	6	LBN:	753240
Count:	48	LBN:	753249
Count:	3	LBN:	754647
Count:	54	LBN:	754734
Count:	24	LBN:	755238
Count:	2046	LBN:	756009
Count:	30	LBN:	758181
Count:	1914	LBN:	758295
Count:	3	LBN:	760881

Count :	3	LBN :	760950
Count :	900	LBN :	760956
Count :	18	LBN :	797724
Count :	138	LBN :	797766
Count :	18	LBN :	807204
Count :	747	LBN :	810546
Count :	7233	LBN :	811350
Count :	7452	LBN :	830586
Count :	15	LBN :	838779
Count :	366	LBN :	838989

Checksum: 57212

\$

Module 13 Labs (Controller Optimizations)

1. Use the program `PERF$:IO_BLASTER` to determine whether your controller/driver performs seek optimizations. Identify the type of optimization performed.

```
$ r io_blaster
Completed I/O  0 on block 1761212
Completed I/O  1 on block 1686009
Completed I/O  2 on block 315254
Completed I/O 14 on block 437218
Completed I/O 12 on block 452528
Completed I/O 19 on block 448599
Completed I/O  8 on block 478056
Completed I/O 16 on block 122624
Completed I/O  6 on block 838662

Completed I/O 13 on block 1032733
Completed I/O  4 on block 1061192
Completed I/O  5 on block 1286529
Completed I/O 18 on block 1384898
Completed I/O 10 on block 1388190
Completed I/O 21 on block 1532757
Completed I/O  3 on block 1576111
Completed I/O 15 on block 1599663
Completed I/O 28 on block 1619440
Completed I/O 17 on block 1642941
Completed I/O 27 on block 1714359
Completed I/O 31 on block 1736759
Completed I/O 29 on block 1788809
Completed I/O 20 on block 1804212
Completed I/O  7 on block 1878055
Completed I/O 24 on block 1900748
Completed I/O 11 on block 1932507
Completed I/O  9 on block 2025165
Completed I/O 23 on block 2026963
Completed I/O 26 on block 887502
Completed I/O 25 on block 233785
Completed I/O 30 on block 181194
Completed I/O 22 on block 120098
$
```

The increasing LBNs followed by decreasing LBNs indicate that a SCAN (elevator seek) optimization is being performed.

Module 14 Labs (Files-11 ODS-2 Concepts)

1. Determine the file open rates and the amount of time being used by the XQP.

\$MONITOR FCP

OpenVMS Monitor Utility
FILE PRIMITIVE STATISTICS
on node ELLIS

28-NOV-1998 12:45:26.35

	CUR	AVE	MIN	MAX
FCP Call Rate	9.69	3.99	0.10	9.69
Allocation Rate	8.19	3.45	0.00	9.50
Create Rate	0.00	0.05	0.00	0.39
Disk Read Rate	0.00	0.07	0.00	0.29
Disk Write Rate	16.29	5.06	0.00	16.29
Volume Lock Wait Rate	0.00	0.00	0.00	0.00
CPU Tick Rate	5.40	1.91	0.00	5.40
File Sys Page Fault Rate	0.00	0.00	0.00	0.00
Window Turn Rate	8.00	3.39	0.00	9.60
File Lookup Rate	1.10	0.37	0.00	1.10
File Open Rate	0.30	0.12	0.00	0.30
Erase Rate	8.19	3.72	0.00	9.39

Examine the "File Open Rate" and "CPU Tick rate". Note: this display shows high Erase, Disk Write, and Allocation rates, indicating that the files being accessed are growing.

Module 15 Labs (Contiguity and the File System)

1. Run the programs PERF\$:sfrag and PERF\$:rfrag on the files created by the fragger program. While they are running, compare the window turn and split transfer rates between the 2 programs. How do they compare? Why?

```
$ spawn/nowait r sfrag
```

```
%DCL-S-SPAWNED, process SYSTEM_1 spawned
```

```
$ monitor/interval=10 io, fcp
```

```
OpenVMS Monitor Utility
```

```
I/O SYSTEM STATISTICS
```

```
on node ELLIS
```

```
28-NOV-1998 13:48:12.70
```

	CUR	AVE	MIN	MAX
Direct I/O Rate	105.90	105.90	105.90	105.90
Buffered I/O Rate	2.40	2.40	2.40	2.40
Mailbox Write Rate	0.00	0.00	0.00	0.00

Note

Very high Split Transfer Rate for sfrag.

Split Transfer Rate	52.20	52.20	52.20	52.20
Log Name Translation Rate	5.50	5.50	5.50	5.50
File Open Rate	0.50	0.50	0.50	0.50
Page Fault Rate	1.90	1.90	1.90	1.90
Page Read Rate	0.00	0.00	0.00	0.00
Page Read I/O Rate	0.00	0.00	0.00	0.00
Page Write Rate	0.00	0.00	0.00	0.00
Page Write I/O Rate	0.00	0.00	0.00	0.00
Inswap Rate	0.00	0.00	0.00	0.00
Free List Size	10469.00	10469.00	10469.00	10469.00
Modified List Size	1283.00	1283.00	1283.00	1283.00

OpenVMS Monitor Utility
 FILE PRIMITIVE STATISTICS
 on node ELLIS
 28-NOV-1998 13:48:22.70

	CUR	AVE	MIN	MAX
FCP Call Rate	0.10	1.14	0.10	2.20
Allocation Rate	0.00	0.00	0.00	0.00
Create Rate	0.00	0.00	0.00	0.00
Disk Read Rate	0.00	0.00	0.00	0.00
Disk Write Rate	0.00	0.14	0.00	0.30
Volume Lock Wait Rate	0.00	0.00	0.00	0.00
CPU Tick Rate	0.00	0.19	0.00	0.40
File Sys Page Fault Rate	0.00	0.00	0.00	0.00

The Window Turn Rate for SFRAG is high.

	CUR	AVE	MIN	MAX
Window Turn Rate	8.00	7.99	8.00	8.00
File Lookup Rate	0.10	0.69	0.10	1.29
File Open Rate	0.00	0.24	0.00	0.50
Erase Rate	0.00	0.00	0.00	0.00

\$ spawn/nowait r rfrag

%DCL-S-SPAWNED, process SYSTEM_1 spawned

\$ monitor/interval=10 io, fcp

OpenVMS Monitor Utility
 I/O SYSTEM STATISTICS
 on node ELLIS
 28-NOV-1998 13:49:19.82

	CUR	AVE	MIN	MAX
Direct I/O Rate	74.69	74.69	74.69	74.69
Buffered I/O Rate	1.79	1.79	1.79	1.79
Mailbox Write Rate	0.10	0.10	0.10	0.10

For rfrag the Split Transfer Rate is high, but not as nearly as high as sfrag.

Split Transfer Rate	7.40	7.40	7.40	7.40
Log Name Translation Rate	5.00	5.00	5.00	5.00
File Open Rate	0.30	0.30	0.30	0.30
Page Fault Rate	0.40	0.40	0.40	0.40
Page Read Rate	0.00	0.00	0.00	0.00
Page Read I/O Rate	0.00	0.00	0.00	0.00
Page Write Rate	0.00	0.00	0.00	0.00
Page Write I/O Rate	0.00	0.00	0.00	0.00
Inswap Rate	0.00	0.00	0.00	0.00
Free List Size	10463.00	10463.00	10463.00	10463.00
Modified List Size	1237.00	1237.00	1237.00	1237.00

OpenVMS Monitor Utility
FILE PRIMITIVE STATISTICS
on node ELLIS
28-NOV-1998 13:49:29.82

	CUR	AVE	MIN	MAX
FCP Call Rate	0.30	0.90	0.30	1.50
Allocation Rate	0.00	0.00	0.00	0.00
Create Rate	0.00	0.00	0.00	0.00
Disk Read Rate	0.00	0.00	0.00	0.00
Disk Write Rate	0.20	0.15	0.10	0.20
Volume Lock Wait Rate	0.00	0.00	0.00	0.00
CPU Tick Rate	0.00	0.10	0.00	0.20
File Sys Page Fault Rate	0.00	0.00	0.00	0.00

Very high Window Turn Rate for rfrag.

Window Turn Rate	69.00	71.05	69.00	73.09
File Lookup Rate	0.10	0.55	0.10	1.00
File Open Rate	0.00	0.15	0.00	0.30
Erase Rate	0.00	0.00	0.00	0.00

\$

These numbers seem to indicate that sfrag is performing sequential access to its fragmented file(s) (the transfers are large and the window is likely turning forward and not back), while, rfrag is performing random access (continually turning the window and probably performing smaller transfers).

Module 16 Labs (File System Caches)

- Determine the file system cache rates.

```
$ monitor/interval=10 file
```

```

                                OpenVMS Monitor Utility
                                FILE SYSTEM CACHING STATISTICS
                                on node ELLIS
                                28-NOV-1998 21:40:19.81

```

		CUR	AVE	MIN	MAX
Dir FCB	(Hit %)	100.00	100.00	0.00	100.00
	(Attempt Rate)	0.10	0.05	0.00	0.10
Dir Data	(Hit %)	100.00	100.00	0.00	100.00
	(Attempt Rate)	0.10	0.05	0.00	0.10
File Hdr	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.05	0.00	0.10
File ID	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00
Extent	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00
Quota	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00
Bitmap	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00

2. Issue the command:

```
$SPAWN/NOWAIT DIR SYS$SYSDEVICE:[000000...]/OUTPUT=NL:
```

Determine the file system cache rates. Repeat adding /FULL to DIR command.

```
$ spawn/nowait dir sys$sysdevice:[000000...]/OUTPUT=NL:
```

```
%DCL-S-SPAWNED, process SYSTEM_1 spawned
```

```
$ monitor/interval=10 file
```

```
OpenVMS Monitor Utility
FILE SYSTEM CACHING STATISTICS
on node ELLIS
28-NOV-1998 21:39:18.52
```

		CUR	AVE	MIN	MAX
Dir FCB	(Hit %)	67.00	67.64	67.00	67.00
	(Attempt Rate)	10.19	10.19	10.19	10.19
Dir Data	(Hit %)	75.00	75.69	75.00	75.00
	(Attempt Rate)	14.39	14.39	14.39	14.39
File Hdr	(Hit %)	100.00	100.00	100.00	100.00
	(Attempt Rate)	6.50	6.50	6.50	6.50
File ID	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00
Extent	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00
Quota	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00
Bitmap	(Hit %)	0.00	0.00	0.00	0.00
	(Attempt Rate)	0.00	0.00	0.00	0.00

Note

The caches being hit are the Dir Data, Dir FCB, and File Hdr, corresponding with the directory lookups.

3. Copy the command procedure PERF\$:CREATE.COM to your directory. Examine the command procedure. Invoke it using: @create/out=create.log. Examine create.log. Compare the file ids and the logical block location of the files in the log. How do they compare? Why?

```
$ @create/out=create.log
$ type/page create.log
Dump of file SYS$SYSROOT:[SYSMGR]ZZ.TOP;1 on 28-NOV-1998 21:44:17.73
File ID (6886,4,0)   End of file block 1 / Allocated 3
```

File Header

```
Header area
  Identification area offset:      40
  Map area offset:                 100
  Access control area offset:      255
  Reserved area offset:           255
  Extension segment number:        0
  Structure level and version:      2, 1
  File identification:              (6886,4,0)
...
Identification area
  File name:                       ZZ.TOP;1
  Revision number:                  1
  Creation date:                   28-NOV-1998 21:44:17.46
  Revision date:                   28-NOV-1998 21:44:17.56
  Expiration date:                 <none specified>
  Backup date:                    <none specified>

Map area
  Retrieval pointers
    Count:      3      LBN:    149541

Checksum:      6661
Dump of file SYS$SYSROOT:[SYSMGR]ZZ.BOTTOM;3 on 28-NOV-1998 21:44:18.36
File ID (6886,5,0)   End of file block 1 / Allocated 3
```

File Header

```
Header area
  Identification area offset:      40
  Map area offset:                 100
  Access control area offset:      255
  Reserved area offset:           255
  Extension segment number:        0
  Structure level and version:      2, 1
  File identification:              (6886,5,0)
```



Note

The ID of this file has the same header number (6886) and the sequence number (5) is one greater. This is due to the reuse of the file id from the ACP_FIDCACHE.

Extension file identification: (0,0,0)
...
Identification area
File name: ZZ.BOTTOM;3
Revision number: 1
Creation date: 28-NOV-1998 21:44:18.07
Revision date: 28-NOV-1998 21:44:18.20
Expiration date: <none specified>
Backup date: <none specified>

Map area
Retrieval pointers
Count: 3 LBN: 149541



Note

The blocks being used are the same as the file previously deleted. This is due to allocation from the ACP_EXTCACHE.

Checksum: 9541
\$

4. Sum up the values for ACP_DINDXCACHE, ACP_DIRCACHE, ACP_HDRCACHE, and ACP_MAPCACHE. Compare the sum against the value for "Maximum Buffers in FCP Cache" in the \$SHOW DEVICE/FULL display for a disk. How do they compare? Why?

```
$ sh dev dka200/full
```

```
Disk ELLIS$DKA200:, device type RZ26L, is online, mounted, file-oriented
device,
    shareable, served to cluster via MSCP Server, error logging is enabled.
```

Error count	0	Operations completed	11188
Owner process	" "	Owner UIC	[SYSTEM]
Owner process ID	00000000	Dev Prot	S:RWPL,O:RWPL,G:R,W
Reference count	284	Default buffer size	512
Total blocks	2050860	Sectors per track	83
Total cylinders	3089	Tracks per cylinder	8

Volume label	"AXP71FT2"	Relative volume number	0
Cluster size	3	Transaction count	373
Free blocks	428022	Maximum files allowed	256357
Extend quantity	5	Mount count	1
Mount status	System	Cache name	"_ELLIS\$DKA200:XQPCACHE"
Extent cache size	64	Maximum blocks in extent cache	42802
File ID cache size	64	Blocks currently in extent cache	11592
Quota cache size	0	Maximum buffers in FCP cache	670
Volume owner UIC	[1,1]	Vol Prot	S:RWCD,O:RWCD,G:RWCD,W:RWCD

```
Volume Status: ODS-2, subject to mount verification, protected subsystems
enabled, file high-water marking, write-through caching enabled.
```

```
$
$ write sys$output f$getsyi("acp_dircache")+f$getsyi("acp_dindxcache")+ -
_$ f$getsyi("acp_hdrcache")+f$getsyi("acp_mapcache")
670
$
```

They should be the same, unless you have shrunken caches.

Module 17 Labs (RMS Structures and Design Considerations)

This module has no formal labs.

Module 18 Labs (RMS Utilities)

1. Use EDIT/FDL to create an FDL file for an indexed file that will contain 10,000 initial records, with 100,000 additional records that are exactly 500 bytes in size, a primary key only that makes up the first 32 bytes. Use \$CREATE/FDL to create a "shell" container file to hold these records.

```
$ edit/fdl/script=indexed dummy.fdl
```

Parsing Definition File

SY\$COMMON:[SYSMGR]DUMMY.IDX; will be created.

Press RETURN to continue (^Z for Main Menu)

Indexed Script

Target disk volume Cluster Size (1-2Giga)[3] :

Number of Keys to Define (1-255)[1] :

Key 0 Graph Type Selection

Line	Bucket Size vs Index Depth	as a 2 dimensional plot
Fill	Bucket Size vs Load Fill Percent	vs Index Depth
Key	Bucket Size vs Key Length	vs Index Depth
Record	Bucket Size vs Record Size	vs Index Depth
Init	Bucket Size vs Initial Load Record Count	vs Index Depth
Add	Bucket Size vs Additional Record Count	vs Index Depth

Graph type to display (Keyword)[Line] :

Number of Records that will be Initially Loaded
into the File (0-2Giga)[-] : 10000

(Fast_Convert NoFast_Convert RMS_Puts)

Initial File Load Method (Keyword)[Fast] :

Number of Additional Records to be Added After
the Initial File Load (0-2147473645)[0] : **100000**

Will Additional Records Typically be Added in
Order by Ascending Primary Key (Yes/No)[No] :

Will Added Records be Distributed Evenly over the
Initial Range of Pri Key Values (Yes/No)[No] :

Key 0 Load Fill Percent (50-100)[100] :

(Fixed Variable)

Record Format (Keyword)[Var] : **fix**

Record Size (1-32224)[-] : **500**

(Bin2 Bin4 Bin8 Int2 Int4 Int8 Decimal String Collated
Dbin2 Dbin4 Dbin8 Dint2 Dint4 Dint8 Ddecimal Dstring Dcollated)

Key 0 Data Type (Keyword)[Str] :

Key 0 Segmentation desired (Yes/No)[No] :

Key 0 Length (1-255)[-] : **32**

Key 0 Position (0-468)[0] :

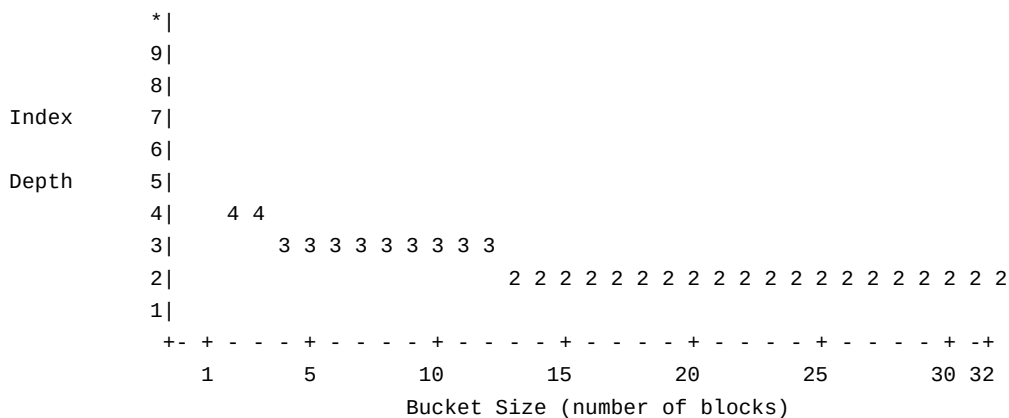
Key 0 Duplicates allowed (Yes/No)[No] :

File Prolog Version (0-3)[3] :

Data Key Compression desired (Yes/No)[Yes] :

Data Record Compression desired (Yes/No)[Yes] :

Index Compression desired (Yes/No)[No] :



```

PV-Prolog Version      3 KT-Key  0 Type      String EM-Emphasis Flatter (15)
DK-Dup Key  0 Values   No KL-Key  0 Length      32 KP-Key  0 Position    0
RC-Data Record Comp    0% KC-Data Key Comp      0% IC-Index Record Comp  0%
BF-Bucket Fill         100% RF-Record Format      Fixed RS-Record Size      500
LM-Load Method Fast_Conv IL-Initial Load      10000 AR-Added Records  100000

```

(Type "FD" to Finish Design)

Which File Parameter (Mnemonic)[refresh] : **FD**

Text for FDL Title Section (1-126 chars)[null]
: **test creation**

Data File file-spec (1-512 chars)[null]
: **dummy.idx**

(Carriage_Return FORTRAN None)
Carriage Control (Keyword)[Carr] :

```

Emphasis Used In Defining Default: ( Flatter_files )
Suggested Bucket Sizes: ( 6 15 24 )
Number of Levels in Index: ( 3 2 2 )
Number of Buckets in Index: ( 634 85 34 )
Pages Required to Cache Index: ( 3804 1275 816 )
Processing Used to Search Index: ( 24 18 20 )

```

Key 0 Bucket Size (2-63)[15] :

Key 0 Name (1-32 chars)[null]
:

Global Buffers desired (Yes/No)[No] :

The Depth of Key 0 is Estimated to be No Greater
than 2 Index levels, which is 3 Total levels.

Press RETURN to continue (^Z for Main Menu)

(Add Delete Exit Help Invoke Modify Quit Set View)
Main Editor Function (Keyword)[Help] : exit

SYS\$SYSROOT:[SYSMGR]DUMMY.IDX;1 43 lines

\$

```
$ type dummy.fdl
TITLE  "test creation"

IDENT  "28-NOV-1998 21:57:59  OpenVMS FDL Editor"

SYSTEM
      SOURCE          "OpenVMS"

FILE
      NAME             "dummy.idx"
      ORGANIZATION     indexed

RECORD
      CARRIAGE_CONTROL carriage_return
      FORMAT           fixed
      SIZE             500
```

AREA 0

ALLOCATION	177423
BEST_TRY_CONTIGUOUS	yes
BUCKET_SIZE	15
EXTENSION	44355

AREA 1

ALLOCATION	1290
BEST_TRY_CONTIGUOUS	yes
BUCKET_SIZE	15
EXTENSION	330

KEY 0

CHANGES	no
DATA_AREA	0
DATA_FILL	100
DATA_KEY_COMPRESSION	yes
DATA_RECORD_COMPRESSION	yes
DUPLICATES	no
INDEX_AREA	1
INDEX_COMPRESSION	no
INDEX_FILL	100
LEVEL1_INDEX_AREA	1
PROLOG	3
SEG0_LENGTH	32
SEG0_POSITION	0
TYPE	string

\$

\$ create/fdl=dummy dummy.idx

\$ dir/full dummy.idx

Directory SYS\$SYSROOT:[SYSMGR]

DUMMY.IDX;2 File ID: (6878,12,0)

Size: 178713/178713 Owner: [SYSTEM]

Created: 28-NOV-1998 22:04:46.59

Revised: 28-NOV-1998 22:05:34.48 (1)

Expires: <None specified>

Backup: <No backup recorded>

Effective: <None specified>

Recording: <None specified>

File organization: Indexed, Prolog: 3, Using 1 key
In 2 areas

Shelved state: Online

Caching attribute: Writethrough

File attributes: Allocation: 178713, Extend: 44355, Maximum bucket size: 15
Global buffer count: 0, No version limit
Contiguous best try

Record format: Fixed length 500 byte records

Record attributes: Carriage return carriage control

RMS attributes: None

Journaling enabled: None

File protection: System:RWED, Owner:RWED, Group:RE, World:

Access Cntrl List: None

Client attributes: None

Total of 1 file, 178713/178713 blocks.

\$

2. Use \$ANALYZE/RMS/FDL to create an FDL file for the SYSUAF.DAT file. Determine the number of index levels in the file.

```
$ ANALYZE/RMS/FDL SYS$SYSTEM:SYSUAF.DAT
$ type/page sysuaf.fdl
IDENT  "28-NOV-1998 22:07:48   OpenVMS ANALYZE/RMS_FILE Utility"

SYSTEM

      SOURCE                      OpenVMS

FILE

      ALLOCATION                    36
      BEST_TRY_CONTIGUOUS         yes
      BUCKET_SIZE                  3
      CLUSTER_SIZE                 3
      CONTIGUOUS                   no
      EXTENSION                     10
      FILE_MONITORING              no
      GLOBAL_BUFFER_COUNT          0
      NAME                         "SYS$COMMON:[SYSEXE]SYSUAF.DAT;1"
      ORGANIZATION                 indexed
      OWNER                        [1,1]
      PROTECTION                   (system:RWED, owner:RWED, group:, world:)

RECORD

      BLOCK_SPAN                   yes
      CARRIAGE_CONTROL             none
      FORMAT                       variable
      SIZE                         1412

AREA 0

      ALLOCATION                    36
      BUCKET_SIZE                  3
      EXTENSION                     10

KEY 0

      CHANGES                     no
      DATA_KEY_COMPRESSION        yes
      DATA_RECORD_COMPRESSION    yes
      DATA_AREA                   0
      DATA_FILL                   100
      DUPLICATES                   no
      INDEX_AREA                   0
      INDEX_COMPRESSION            yes
      INDEX_FILL                   100
      LEVEL1_INDEX_AREA           0
      NAME                         ""
      NULL_KEY                     no
      PROLOG                       3
      SEG0_LENGTH                  32
      SEG0_POSITION                4
      TYPE                         string
```

KEY 1

CHANGES	yes
DATA_KEY_COMPRESSION	no
DATA_AREA	0
DATA_FILL	100
DUPLICATES	yes
INDEX_AREA	0
INDEX_COMPRESSION	no
INDEX_FILL	100
LEVEL1_INDEX_AREA	0
NAME	""
NULL_KEY	no
SEG0_LENGTH	4
SEG0_POSITION	36
TYPE	bin4

KEY 2

CHANGES	yes
DATA_KEY_COMPRESSION	no
DATA_AREA	0
DATA_FILL	100
DUPLICATES	yes
INDEX_AREA	0
INDEX_COMPRESSION	no
INDEX_FILL	100
LEVEL1_INDEX_AREA	0
NAME	""
NULL_KEY	no
SEG0_LENGTH	8
SEG0_POSITION	36
TYPE	bin8

KEY 3

CHANGES	yes
DATA_KEY_COMPRESSION	no
DATA_AREA	0
DATA_FILL	100
DUPLICATES	yes
INDEX_AREA	0
INDEX_COMPRESSION	no
INDEX_FILL	100
LEVEL1_INDEX_AREA	0
NAME	""
NULL_KEY	no
SEG0_LENGTH	8
SEG0_POSITION	44
TYPE	bin8

ANALYSIS_OF_AREA	0
RECLAIMED_SPACE	0

```
ANALYSIS_OF_KEY 0
  DATA_FILL            86
  DATA_KEY_COMPRESSION 63
  DATA_RECORD_COMPRESSION 64
  DATA_RECORD_COUNT    11
  DATA_SPACE_OCCUPIED   6
```

Key 0 has an index depth of 1.

```
  DEPTH                1
  INDEX_COMPRESSION     67
  INDEX_FILL            2
  INDEX_SPACE_OCCUPIED  3
  LEVEL1_RECORD_COUNT   2
  MEAN_DATA_LENGTH      644
  MEAN_INDEX_LENGTH     34
```

```
ANALYSIS_OF_KEY 1
  DATA_FILL            8
  DATA_KEY_COMPRESSION 0
  DATA_RECORD_COUNT    11
  DATA_SPACE_OCCUPIED   3
```

Key 1 has an index depth of 1.

```
  DEPTH                1
  DUPLICATES_PER_SIDR  0
  INDEX_COMPRESSION     0
  INDEX_FILL            1
  INDEX_SPACE_OCCUPIED  3
  LEVEL1_RECORD_COUNT   1
  MEAN_DATA_LENGTH      11
  MEAN_INDEX_LENGTH     6
```

```
ANALYSIS_OF_KEY 2
  DATA_FILL            11
  DATA_KEY_COMPRESSION 0
  DATA_RECORD_COUNT    11
  DATA_SPACE_OCCUPIED   3
```

Key 2 has an index depth of 1.

DEPTH	1
DUPLICATES_PER_SIDR	0
INDEX_COMPRESSION	0
INDEX_FILL	1
INDEX_SPACE_OCCUPIED	3
LEVEL1_RECORD_COUNT	1
MEAN_DATA_LENGTH	15
MEAN_INDEX_LENGTH	10

ANALYSIS_OF_KEY 3	
DATA_FILL	5
DATA_KEY_COMPRESSION	0
DATA_RECORD_COUNT	1
DATA_SPACE_OCCUPIED	3

Key 3 has an index depth of 1.

DEPTH	1
DUPLICATES_PER_SIDR	10
INDEX_COMPRESSION	0
INDEX_FILL	1
INDEX_SPACE_OCCUPIED	3
LEVEL1_RECORD_COUNT	1
MEAN_DATA_LENGTH	66
MEAN_INDEX_LENGTH	10

\$

Module 19 Labs (RMS I/O Buffering Considerations)

1. Determine the elapsed times and I/Os when running `PERF$ssfrag` to read the files generated by `fragger`. Determine the number of I/Os and the elapsed time required to run the program. Run the program with multi-block counts of 32, 64, and 96 blocks. Compare the elapsed times and I/O counts for the sample runs.

```
$ SH RMS
MULTI- |          MULTIBUFFER COUNTS          | NETWORK
BLOCK  | Indexed Relative Sequential          | BLOCK
COUNT |          Disk  Magtape  Unit Record | COUNT
Process 0 |    0    0    0    0    0    0    0 |    0
System 16 |    0    0    0    0    0    0    0 |    8

          Prolog  Extend Quantity  VCC_DFW
Process 0          0          0          0
System 0          0          0          0

$ show status
Status on 28-NOV-1998 22:11:11.75 Elapsed CPU : 0 00:00:17.41
Buff. I/O : 3141 Cur. ws. : 1424 Open files : 0
Dir. I/O : 816 Phys. Mem. : 1264 Page Faults : 5627

$ r ssfrag
$ show status
Status on 28-NOV-1998 22:12:10.00 Elapsed CPU : 0 00:00:32.95
Buff. I/O : 3170 Cur. ws. : 1424 Open files : 0
Dir. I/O : 7093 Phys. Mem. : 1296 Page Faults : 5694

$
```

\$ SET RMS/BLOCK=32

\$ show status

Status on 28-NOV-1998 22:12:22.48 Elapsed CPU : 0 00:00:33.05
Buff. I/O : 3189 Cur. ws. : 1424 Open files : 0
Dir. I/O : 7093 Phys. Mem. : 1328 Page Faults : 5750

\$ r ssfrag

\$ show status

Status on 28-NOV-1998 22:13:13.95 Elapsed CPU : 0 00:00:45.56
Buff. I/O : 3230 Cur. ws. : 1424 Open files : 0
Dir. I/O : 10227 Phys. Mem. : 1376 Page Faults : 5802

\$

\$ SET RMS/BLOCK=64

\$ show status

Status on 28-NOV-1998 22:13:38.79 Elapsed CPU : 0 00:00:45.70
Buff. I/O : 3256 Cur. ws. : 3824 Open files : 0
Dir. I/O : 10227 Phys. Mem. : 1248 Page Faults : 5846

\$ r ssfrag

\$ show status

Status on 28-NOV-1998 22:14:18.48 Elapsed CPU : 0 00:00:53.89
Buff. I/O : 3297 Cur. ws. : 3824 Open files : 0
Dir. I/O : 11797 Phys. Mem. : 1232 Page Faults : 5921

\$

\$ set rms/block=96

\$ show status

Status on 28-NOV-1998 22:15:06.66 Elapsed CPU : 0 00:00:54.00
Buff. I/O : 3323 Cur. ws. : 1424 Open files : 0
Dir. I/O : 11797 Phys. Mem. : 1312 Page Faults : 5977

\$ r ssfrag

\$ show status

Status on 28-NOV-1998 22:15:44.27 Elapsed CPU : 0 00:01:01.71
Buff. I/O : 3364 Cur. ws. : 1424 Open files : 0
Dir. I/O : 12847 Phys. Mem. : 1264 Page Faults : 6039

Multiblock Count	Elapsed Time	Direct I/O
16	58.25	6277
32	51.47	3134
64	39.69	1570
96	37.61	1050

Module 20 Labs (\$MONITOR RMS)

1. Enable file statistics on the files X.DAT and Y.DAT. Determine the types of file operations performed on these files using the program PERF\$:sfrag.exe and PERF\$:rfrag.exe.

```
$ set file/stat x.dat,y.dat
$ spawn/nowait r sfrag
%DCL-S-SPAWNED, process SYSTEM_1 spawned
$
$ MONITOR RMS/FILE=(X.DAT,Y.DAT)/ITEM=OPER
```

OpenVMS Monitor Utility					
RMS FILE OPERATIONS					
on node ELLIS					
28-NOV-1998 22:18:20.16					
(Sequential) ELLIS\$DKA200:[SYS0.][SYSMGR]Y.DAT;2					
Active Streams:	1	CUR	AVE	MIN	MAX
\$GET Call Rate	(Seq)	881.66	875.87	842.66	917.33
	(Key)	0.00	0.00	0.00	0.00
	(RFA)	0.00	0.00	0.00	0.00
\$FIND Call Rate	(Seq)	0.00	0.00	0.00	0.00
	(Key)	0.00	0.00	0.00	0.00
	(RFA)	0.00	0.00	0.00	0.00
\$PUT Call Rate	(Seq)	0.00	0.00	0.00	0.00
	(Key)	0.00	0.00	0.00	0.00
\$READ Call Rate		0.00	0.00	0.00	0.00
\$WRITE Call Rate		0.00	0.00	0.00	0.00
\$UPDATE Call Rate		0.00	0.00	0.00	0.00
\$DELETE Call Rate		0.00	0.00	0.00	0.00
\$TRUNCATE Call Rate		0.00	0.00	0.00	0.00
\$EXTEND Call Rate		0.00	0.00	0.00	0.00
\$FLUSH Call Rate		0.00	0.00	0.00	0.00

SFRAG is performing sequential reads as we anticipated based on the split transfer rate.

```
$ spawn/nowait r rfrag
$ MONITOR RMS/FILE=(X.DAT,Y.DAT)/ITEM=OPER
```

```

                                OpenVMS Monitor Utility
                                RMS FILE OPERATIONS
                                on node ELLIS
                                28-NOV-1998 22:20:37.04
(Sequential) ELLIS$DKA200:[SYS0.][SYSMGR]X.DAT;2
Active Streams:  1              CUR      AVE      MIN      MAX

$GET Call Rate   (Seq)         0.00     0.00     0.00     0.00
                  (Key)         0.00     0.00     0.00     0.00
                  (RFA)        37.66    37.66    37.66    37.66
$FIND Call Rate  (Seq)         0.00     0.00     0.00     0.00
                  (Key)         0.00     0.00     0.00     0.00
                  (RFA)         0.00     0.00     0.00     0.00
$PUT Call Rate   (Seq)         0.00     0.00     0.00     0.00
                  (Key)         0.00     0.00     0.00     0.00
$READ Call Rate          0.00     0.00     0.00     0.00
$WRITE Call Rate         0.00     0.00     0.00     0.00
$UPDATE Call Rate        0.00     0.00     0.00     0.00
$DELETE Call Rate        0.00     0.00     0.00     0.00
$TRUNCATE Call Rate       0.00     0.00     0.00     0.00
$EXTEND Call Rate        0.00     0.00     0.00     0.00
$FLUSH Call Rate         0.00     0.00     0.00     0.00

```

RFRAG is performing RFA (random) reads, as we anticipated with its high window turn rate.

Module 21 Labs (Understanding CPU Performance)

This module has no formal labs.

Module 22 Labs (Understanding Time Spent in Modes)

- Determine where CPU time is being spent, when running the programs:

- loc2 with a working set size of 8000

```
$ set work/lim=8000/quo=8000/ext=8000
```

```
%SET-I-NEWLIMS, new working set: Limit = 2848 Quota = 2848 Extent = 8000
```

```
$ spawn/nowait r loc2
```

```
$ MONITOR MODES
```

```

OpenVMS Monitor Utility
TIME IN PROCESSOR MODES
on node ELLIS
28-NOV-1998 22:50:31.75

+-----+
| CUR |
+-----+

0          25          50          75          100
+ - - - - + - - - - + - - - - + - - - - +
Interrupt State 2 |
|
|          |          |          |          |
MP Synchronization
|
|          |          |          |          |
Kernel Mode 93 | *****
|
|          |          |          |          |
Executive Mode
|
|          |          |          |          |
Supervisor Mode
|
|          |          |          |          |
User Mode 6 | **
|
|          |          |          |          |
Compatibility Mode
|
|          |          |          |          |
Idle Time
+ - - - - + - - - - + - - - - + - - - - +

```

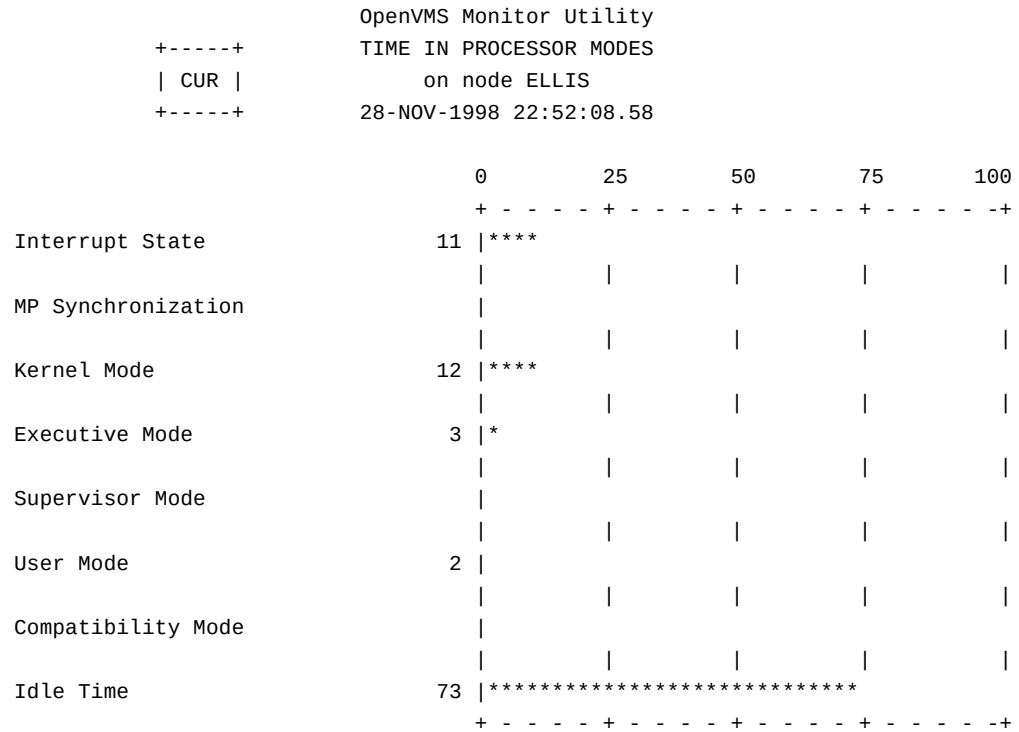
e. fragger

```
$ spawn/nowait r fragger
```

```
%DCL-S-SPAWNED, process SYSTEM_1 spawned
```

```
$ mon modes
```

```
$ mon modes/int=10
```



f. sfrag

```
$ spawn/nowait r sfrag
$ mon modes/int=10
```

```

                                OpenVMS Monitor Utility
                                TIME IN PROCESSOR MODES
                                on node ELLIS
                                28-NOV-1998 22:53:31.86

0          25          50          75          100
+-----+
| CUR |
+-----+

Interrupt State      22 |*****
|                    |
MP Synchronization  |
|                    |
Kernel Mode         16 |*****
|                    |
Executive Mode      21 |*****
|                    |
Supervisor Mode     |
|                    |
User Mode           4 |*
|                    |
Compatibility Mode  |
|                    |
Idle Time           38 |*****
+-----+
+-----+ +-----+ +-----+ +-----+ +-----+

```

Module 23 Labs (OpenVMS Scheduling and Priorities)

1. Run the program loc2 in the background 3 times with a default QUANTUM setting and working set sizes of 8000.

Do the same with QUANTUM set to 2. Do you notice any difference in response time?

You should probably be noticing better response time with QUANTUM set to 2.

2. Determine the process scheduling states on the system?

```
$ mon state/int=10
```

```

                                OpenVMS Monitor Utility
                                PROCESS STATES
                                on node ELLIS
                                29-NOV-1998 00:32:54.18

                                0          10          20          30          40
                                + - - - - + - - - - + - - - - + - - - - +
Collided Page Wait              |
Mutex & Misc Resource Wait      |
Common Event Flag Wait          |
Page Fault Wait                 |
Local Event Flag Wait           6 |*****
Local Evt Flg (Outswapped)      |
                                |          |          |          |
Hibernate                      22 |*****
Hibernate (Outswapped)          |
Suspended                      |
Suspended (Outswapped)          |
Free Page Wait                  |
Compute                        |
Compute (Outswapped)            |
Current Process                 1 |*
                                + - - - - + - - - - + - - - - + - - - - +

```

Module 24 Labs (PIXSCAN and DORMANTWAIT)

1. Run looper in the background at priority 1. Run it again in the background at priority 4. Using \$MONITOR PROCESS/TOPCPU determine whether the priority 1 process gets any CPU time. If so, why?

```
$ run/process=looper/priority=1 looper
%RUN-S-PROC_ID, identification of created process is 20200129
$ run/process=blooper/priority=4 looper
%RUN-S-PROC_ID, identification of created process is 2020012A
$ monitor process/topcpu/int=10

OpenVMS Monitor Utility
TOP CPU TIME PROCESSES
on node ELLIS
29-NOV-1998 00:37:25.04
```

		0	25	50	75	100												
		+	-	-	-	+	-	-	-	+	-	-	-	+	-	-	-	+
2020012A	BL00PER	72	*****															
2020011E	DECW\$SERVER_0	16	*****															
20200129	LOOPER	5	**															
...																		
						+	-	-	-	+	-	-	-	+	-	-	-	+

The priority 1 process receives CPU time due to PIXSCAN boosts.

Module 25 Labs (Locking Concepts)

1. Determine the current number of locks and resources on the system.

\$ monitor lock

OpenVMS Monitor Utility
LOCK MANAGEMENT STATISTICS
on node ELLIS
29-NOV-1998 00:39:09.33

	CUR	AVE	MIN	MAX
New ENQ Rate	0.00	0.50	0.00	1.00
Converted ENQ Rate	0.00	0.16	0.00	0.33
DEQ Rate	0.00	0.50	0.00	1.00
Blocking AST Rate	0.00	0.00	0.00	0.00
ENQs Forced To Wait Rate	0.00	0.00	0.00	0.00
ENQs Not Queued Rate	0.00	0.00	0.00	0.00
Deadlock Search Rate	0.00	0.00	0.00	0.00
Deadlock Find Rate	0.00	0.00	0.00	0.00
Total Locks	1177.00	1177.00	1177.00	1177.00
Total Resources	1161.00	1161.00	1161.00	1161.00

There are currently 1177 Locks and 1161 Resources on the system.

Module 26 Labs (Distributed Locking Concepts)

Optional labs may be offered dependent on configuration.

Module 27 Labs (MSCP Server)

This module has no formal labs.