RMI
A Programmers Guide To Performance

peter ranisch
OpenVMS@ranisch.at

SYS$GETRMI

Documentation:

HPOpenVMSSystem Services
Reference Manual: A–GETUAI

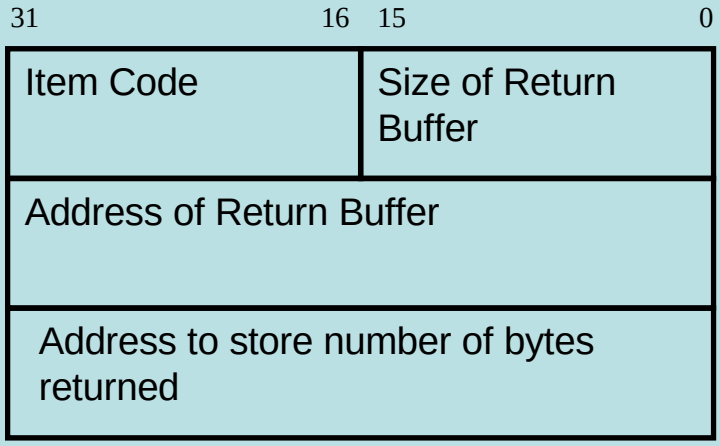
OpenVMS 7.3-2 page SYS1-727
OpenVMS 8.2 page SYS1-736

Features

- Standard system service interface to obtain performance management data
- All MONITOR utility data items are available
- Close to 300 items are available.
- Some items returned are not currently updated by OpenVMS
- Uses an item list (ile3) format

2006-05-17 OpenVMS@ranisch.at 3

ile3



2006-05-17 OpenVMS@ranisch.at 4

Simple_getrmi.c

```
$ type simple_getrmi.c
/* Sample call to SYS$GETRMI.
   Author:  Bruce Ellis, BRUDEN Corporation
*/
#include <stdio.h>
#include <starlet.h>
#include <ssdef.h>
#include <stdlib.h>
#include <iosbdef.h>
```

2006-05-17 OpenVMS@ranisch.at 5

Simple_getrmi.c

```
/* RMI symbol definitions. */
#include <rmidef.h>
#include <string.h>
/* Item List Entry definitions. */
#include <iledef.h>
int sys$getrmi (unsigned int , unsigned int ,
   unsigned int,
       void *, struct _iosb *, void
   (*)(__unknown_params), int);
```

2006-05-17 OpenVMS@ranisch.at 6

Simple_getrmi.c

```
#define check(S) if(!((S)&1)) sys$exit(S)
#define RMI_EFN 33
int     main(void)
{
        int     status;
        iosb    ios;
        int     num_com;
        int     num_cur;
```


Simple_getrmi.c

```
/* Set up item list for the call. */
      ile3   items[ ] =
  {{sizeof(num_com),RMI$_COM,&num_co
  m},

  {sizeof(num_cur),RMI$_CUR,&num_cur},
                {0,0}};
/* Call sys$getrmi. */
      status =
  sys$getrmi(RMI_EFN,0,0,items,&ios,0,0);
        check(status);
        // wait for the RMI completion
        status = sys$waitfr(RMI_EFN);
```

Simple_getrmi.c

```
        check(status);
        check(ios.iosb$w_status);
/* Display the data. */
        printf("COM: %10d\n",num_com);
        printf("CUR: %10d\n",num_cur);
        return(SS$_NORMAL);
}
```

2006-05-17 OpenVMS@ranisch.at 9

Simple_getrmi.c

```
$ cc simple_getrmi
$ link simple_getrmi
$ run  simple_getrmi
COM:          0
CUR:          1
$
```

2006-05-17 OpenVMS@ranisch.at 10

Wait States

Mnemonic	Voluntary	Name	Reason to Enter	Reason to Leave
HIB	Yes	Hibernating	Process hibernates.	Process is woken by another process, scheduled a wake up that has expired, or wakes itself.
LEF	Yes	Local Event Flag	Process is waiting for an event to complete (whose completion is signaled by setting a local event flag). Normal events associated with LEF states include: I/O request, Timer request, or Lock request.	I/O completes, timer expires, or lock is granted.
CEF	Yes	Common Event Flag	Process is waiting for another process to set a common event flag.	Other process sets the common event flag.
SUSP	Yes	Suspended	Process is suspended by another process ($SET PROCESS/SUSPEND /IDENTIFICATION=*pid*).	Process is resumed by another process ($SET PROCESS /RESUME /IDENTIFICATION=*pid*).
PFW	No	Page Fault Wait	Process is waiting to complete the read associated with a hard page fault.	The page read completes.
COLPG	No	Collided Page	Process enters PFW state on a page that already has a process in PFW state on it.	The page read completes.
FPG	No	Free Page Wait	Process generates a hard fault or a demand zero fault and no pages are on the free page list to support the fault.	The free page list is replenished and the fault is resolved.
MWAIT	No	Misc. Resource or MUTEX wait	Process is waiting on a miscellaneous resource or a Mutual Exclusion Semaphore (MUTEX)	The resource or MUTEX becomes available.

2006-05-17 OpenVMS@ranisch.at 11

GETRMI CPU States Items

Item	Meaning	Concern
RMI$_CEF	Number of processes in common event flag wait state.	Not generally a performance concern.
RMI$_COLPG	Number of processes in collided page wait state.	Memory depletion. Oversized working sets.
RMI$_COM	Number of processes in computable state.	CPU contention. Response time degradation.
RMI$_COMO	Number of processes in computable outswapped state.	Memory exhaustion. Possibly page/swap file exhaustion.
RMI$_CUR	Number of current processes.	Not generally a performance concern, unless there are computable processes and number of current processes is less than number of CPUs.

2006-05-17 OpenVMS@ranisch.at 12

GETRMI CPU States Items

RMI$_FPG	Number of processes in free page wait state.	Complete memory exhaustion. Possibly, page/swap file exhaustion.
RMI$_HIB	Number of processes in hibernation wait state.	Not generally a performance issue.
RMI$_HIBO	Number of processes in hibernation outswapped state.	Memory exhaustion.
RMI$_LEF	Number of processes in local event flag wait state.	Not generally a performance issue. May be caused by lock contention.
RMI$_LEFO	Number of processes in local event flag outswapped wait state.	Memory exhaustion.

2006-05-17 OpenVMS@ranisch.at 13

GETRMI CPU States Items

RMI$_MWAIT	Number of processes in mutex/ miscellaneous resource wait state.	System resource/quota exhaustion. Possible code problem. (SHOW SYSTEM/SDA/AM give interpretations of MWAIT processes.
RMI$_PFW	Number of processes in page fault wait state.	Memory exhaustion. Thrashing. NOTE: currently returns number of hibernating processes.
RMI$_SUSP	Number of processes in suspended wait state.	Not generally a performance issue.
RMI$_SUSPO	Number of processes in suspended outswapped wait state.	Memory exhaustion.

2006-05-17 OpenVMS@ranisch.at 14

GETRMI CPU Mode Information

RMI$_CPUEXEC	Executive mode time.	Investigate Record Management Services/ Database operations
RMI$_CPUIDLE	Idle time	Some idle time is good for response-time oriented systems. For throughput oriented systems, should be close to zero.
RMI$_CPUINTSTK	Interrupt stack/state time	Investigate explicit and implicit (distributed locking/MSCP serving) I/O. Take advantage of “fast path” processing.

2006-05-17 OpenVMS@ranisch.at 15

GETRMI CPU Mode Information

RMI$_CPUKERNEL	Kernel mode time.	Paging I/O File System Operations Locking In general, look for inefficient use of system.
RMI$_CPUMPSYNCH	MP Synchronization time	Tune activities that have impact on kernel mode and interrupt state time. Make it a goal to stay at the most recent release of OpenVMS
RMI$_CPUSUPER	Supervisor mode time	Generally not high. If excessive, evaluate use of command procedures on the system. Possibly, database package.

2006-05-17 OpenVMS@ranisch.at 16

GETRMI CPU Mode Information

RMI$_CPUUSER	User mode time	Tune applications. Consider more CPUs. Consider a lower quantum.
RMI$_CPUCOMPAT	Compatibility mode	PDP-11 applications – time to migrate

2006-05-17 OpenVMS@ranisch.at 17

GETRMI CPU Mode Information

- All datas are in 10msec multiples
- All CPUs, no per CPU data
- 64-bit variables
- Since begin of system

2006-05-17 OpenVMS@ranisch.at 18

CPU Mode sample

```
__int64 int_state;
__int64 mp_synch;
__int64 kernel;
__int64 exec;
__int64 super;
__int64 user;
__int64 idle;
__int64 total;
static __int64 oint_state=0;
static __int64 omp_synch=0;
static __int64 okernel=0;
static __int64 oexec=0;
static __int64 osuper=0;
static __int64 ouser=0;
static __int64 oidle=0;
static __int64 ototal = 0;
static __int64 dtotal = 0;
```

2006-05-17 OpenVMS@ranisch.at 19

CPU Mode sample

```
ile3    items[] = {{sizeof(int_state),RMI$_CPUINTSTK,&int_state},
    {sizeof(mp_synch),RMI$_CPUMPSYNCH,&mp_synch},
    {sizeof(kernel),RMI$_CPUKERNEL,&kernel},
    {sizeof(exec),RMI$_CPUEXEC,&exec},
    {sizeof(super),RMI$_CPUSUPER,&super},
    {sizeof(user),RMI$_CPUUSER,&user},
    {sizeof(idle),RMI$_CPUIDLE,&idle},
 {0,0}};

 // Get the CPU modes
       status = sys$getrmi(RMI_EFN,0,0,items,&ios,0,0);
        check(status);
// wait for the RMI completion
        status = sys$waitfr(RMI_EFN);
        check(status);
        check(ios.iosb$w_status);
```

2006-05-17 OpenVMS@ranisch.at 20

CPU Mode sample

```
// CPU mode stats
      total = int_state+mp_synch+kernel+exec+super+user+idle;
      dtotal = total-ototal;
      printf("%7s %7s %7s %7s %7s %7s
   %7s\n","IState","MPSync","Kernel",
            "Exec","Super","User","Idle");
      printf("%6.2lf%% %6.2lf%% %6.2lf%% %6.2lf%% "
            " %6.2lf%% %6.2lf%% %6.2lf%%\n",
            (double)(int_state-oint_state)*100.0/dtotal,
            (double)(mp_synch-omp_synch)*100.0/dtotal,
            (double)(kernel-okernel)*100.0/dtotal,
            (double)(exec-oexec)*100.0/dtotal,
            (double)(super-osuper)*100.0/dtotal,
            (double)(user-ouser)*100.0/dtotal,
            (double)(idle-oidle)*100.0/dtotal
            );
```


CPU Mode sample

```
oint_state = int_state;
omp_synch = mp_synch;
okernel = kernel;
oexec = exec;
osuper = super;
ouser = user;
oidle = idle;
ototal = total;
```

RMI$_MSCP_EVERYTHING

Returns all the performance data items in the following order:

MSCP_BUFAVL	Current number of free MSCP buffers
MSCP_BUFSMALL	Smallest MSCP buffer size allowed
MSCP_BUFWAITCUR	Number of requests currently queued waiting for MSCP buffer memory

2006-05-17 OpenVMS@ranisch.at 23

RMI$_MSCP_EVERYTHING continued

MSCP_BUFWAITPEAK	Maximum number of requests simultaneously queued waiting for MSCP buffer
MSCP_DSKSRV	Number of MSCP served disks
MSCP_HSTSRV	Number of MSCP served hosts
MSCP_LB_FAILCNT	MSCP server ’s count of failed load-balancing requests
MSCP_LB_INITCNT	MSCP server ’s count of load-balancing requests sent

2006-05-17 OpenVMS@ranisch.at 24

RMI$_MSCP_EVERYTHING continued

MSCP_LB_LMLOAD1	MSCP server 's previous interval's load 1 value
MSCP_LB_LMLOAD2	MSCP server 's previous interval's load 2 value
MSCP_LB_LMLOAD3	MSCP server 's previous interval's load 3 value
MSCP_LB_LMLOAD4	MSCP server 's previous interval's load 4 value

2006-05-17 OpenVMS@ranisch.at 25

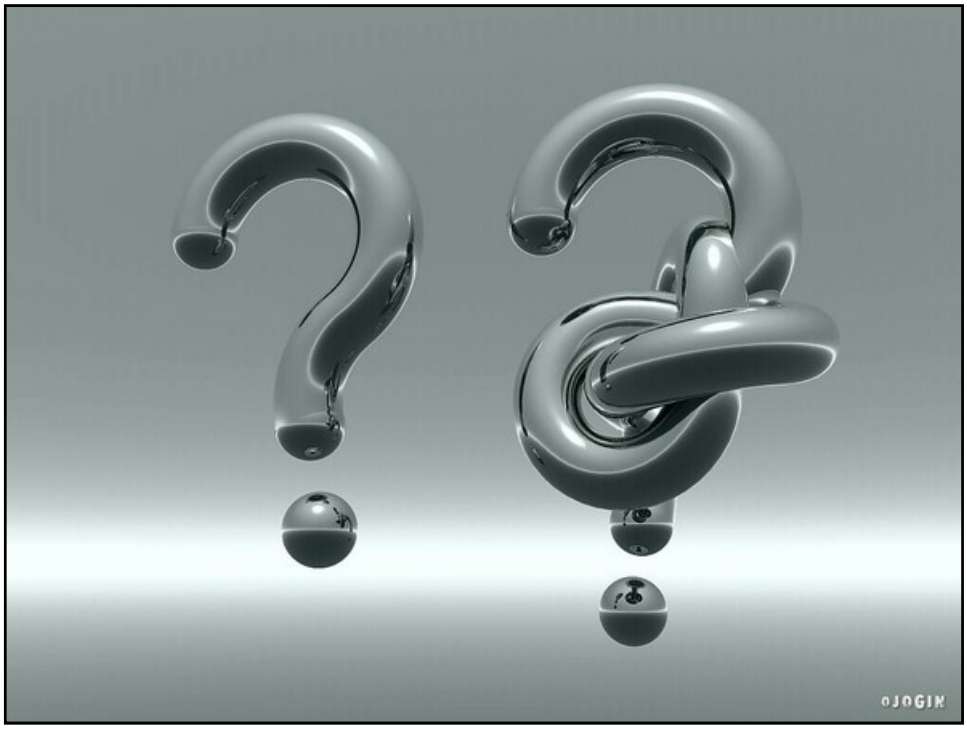
RMI$_MSCP_EVERYTHING continued

MSCP_LB_LOAD	MSCP server 's target load for load-balancing requests
MSCP_LB_LOAD_AVAIL	MSCP server 's current load available value
MSCP_LB_LOAD_CAP	MSCP server 's load capacity value

.....

Because this an array of **35 longwords**, the buffer length field in the item descriptor should specify 4 times 35 (bytes).

2006-05-17 OpenVMS@ranisch.at 26


OJOGIN