OpenVMS Technical Journal V15

Class Scheduling on OpenVMS

Shriniketan Bhagwat

Executive summary

This article outlines the OpenVMS class scheduling, its usage, its advantages, and its internal data structures.
The following topics are covered:

- Introduction to class scheduling
- Class scheduling advantages
- Class scheduler database file
- In-memory data structures

Introduction to OpenVMS class scheduling

The class scheduler provides the ability to limit the amount of CPU time that a system's users may receive by placing the users into scheduling classes. Each class is assigned a percentage of the overall system's CPU time. As the system runs, the combined sets of users in a class are limited to the percentage of CPU execution time allocated to their class. To invoke the class scheduler, use the SYSMAN interface.

Class scheduling is implemented in the SYSMAN utility, which allows users to define classes based on username, UIC, or account. SYSMAN allows users to create, delete, modify, suspend, resume, and display scheduling classes.

Creating a scheduling class

The CLASS_SCHEDULE ADD command creates a scheduling class. The highlights of the CLASS_SCHEDULE ADD command include:

- Identifies users in the class, by account name, user name, or UIC
- Specifies the percent of CPU time allotted to processes that are run by users in this scheduling class on primary days and secondary days and specifies the hourly ranges during which the CPU time restriction applies
- Allows a scheduling class to receive additional CPU time when the CPU is idle.

Note:
Creating the scheduling class does not affect a process already running. Adding the scheduling class through SYSMAN requires OPER privilege.

Example:

```
SYSMAN> CLASS_SCHEDULE ADD MAINCLASS -
_SYSMAN> /ACCOUNT = (ACCTNAME1, ACCTNAME2) -
_SYSMAN> /USERNAME = HOTSHOT -
_SYSMAN> /CPULIMIT = (PRIMARY, 08-17=15, SECONDARY, 00-23=30) -
_SYSMAN> /WINDFALL
```

Deleting a scheduling class

The CLASS_SCHEDULE DELETE command deletes the scheduling class from the class scheduler database file. The highlights of the CLASS_SCHEDULE DELETE command include:

- All processes that are members of this scheduling class are no longer class scheduled.
- A scheduling class cannot be deleted if the scheduling class is active, where active is represented by any of the users who belong to the class having a logged-in status

Note: Deleting the scheduling class through SYSMAN requires OPER privilege.

Example:

```
SYSMAN> CLASS_SCHEDULE DELETE MAINCLASS
```

Modifying a scheduling class

The CLASS_SCHEDULE MODIFY command changes the characteristics of a scheduling class.

Note:
Modifying the scheduling class through SYSMAN requires OPER privilege.

Example:

```
SYSMAN> CLASS_SCHEDULE MODIFY MAINCLASS -
_SYSMAN> /ACCOUNT = (ACCTNAME1, ACCTNAME2) -
_SYSMAN> /USERNAME = HOTSHOT -
_SYSMAN> /CPU_LIMIT = (PRIMARY, 08-13=20, SECONDARY, 00-20=40) -
_SYSMAN> /NOWINDFALL
```

Displaying a scheduling class

The CLASS_SCHEDULE SHOW command displays information about a scheduling class.

Note:
Displaying the scheduling class through SYSMAN requires OPER privilege.

Example:

```
SYSMAN> CLASS_SCHEDULE SHOW MAINCLASS /FULL
```

Suspending a scheduling class

The CLASS_SCHEDULE SUSPEND command suspends the specified scheduling class. All processes that are part of the scheduling class remain as part of the scheduling class but are granted unlimited CPU time.

Note:
Suspending the scheduling class does not affect a process already running. Suspending the scheduling class through SYSMAN requires OPER privilege.

Example:

```
SYSMAN> CLASS_SCHEDULE SUSPEND MAINCLASS
```

Resuming a scheduling class

The CLASS_SCHEDULE RESUME command resumes a scheduling class that is currently suspended.

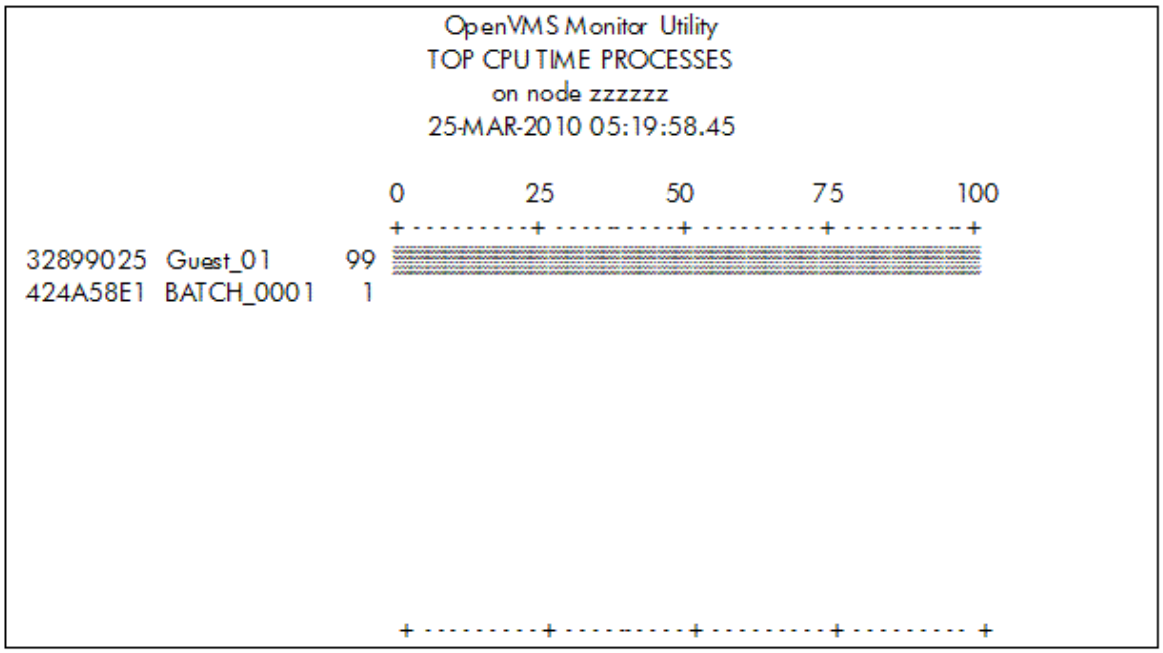
Note:
Resuming the scheduling class does not affect a process already running. Resuming the scheduling class through SYSMAN requires OPER privilege.

Example:

```
SYSMAN> CLASS_SCHEDULE RESUME MAINCLASS
```

Advantages of class scheduling

The following example explains the use of the class scheduling. The user has installed a third-party application for example, ABC over which the user does not have any control. When ABC is started, assume that it consumes an entire CPU's processing power and impacts other applications on the system. The output of $ MONITOR PROCESS/TOPCPU is similar to the following:

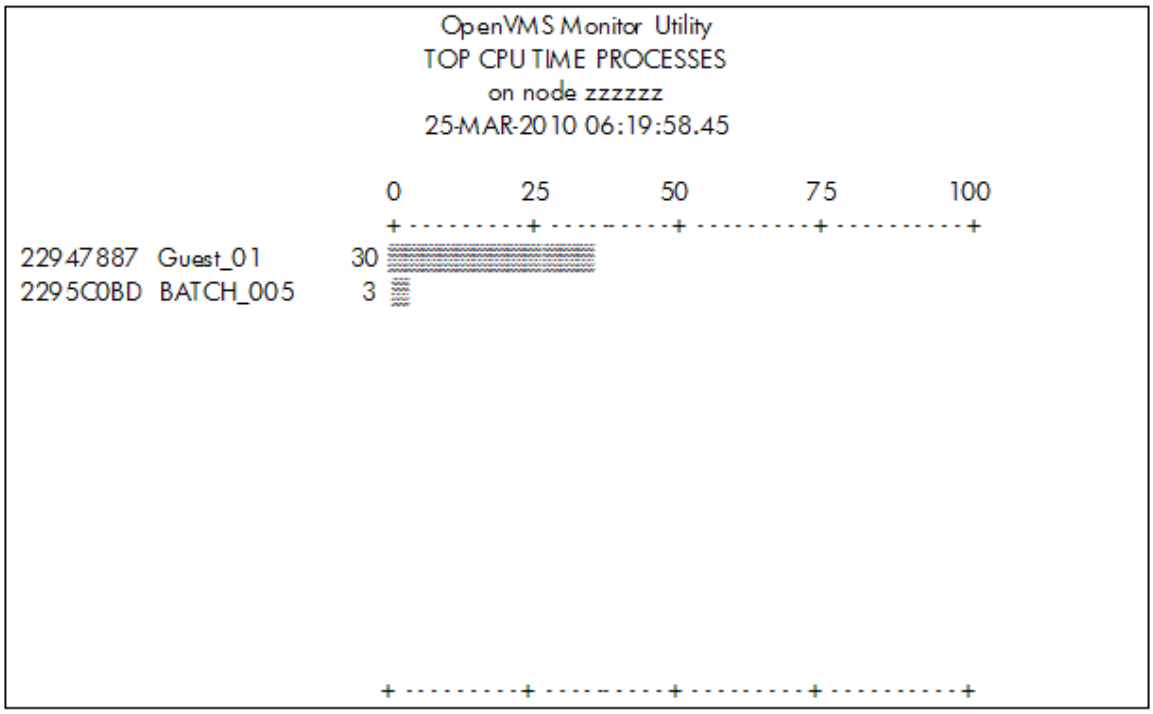


To overcome this problem, create a class to limit the amount of CPU the processes can consume. The process is running under the username "GUEST". You can create a class to limit the CPU for this username to 30 percent during working hours, Monday to Friday, and allow the user free reign on the weekends and any time outside the primary hours specified:

```
$ mcr sysman
SYSMAN> class add limit_process/cpulimit=(primary, 8-17=30)/user=guest
SYSMAN> class show/all/full

Node zzzzzz:
------------
Class name:    LIMIT_PROCESS              Windfall:    Disabled

Maximum CPU time for Primary Days:
                30%  ( 8am - 5pm )
Maximum CPU time for Secondary Days:
                None
Primary Days:    Mon  Tue  Wed  Thu  Fri

Secondary Days:                   Sat  Sun

Nodes:
      zzzzzz
User names:
      GUEST
```

Now, when the application ABC is run from the GUEST account during the day, you can see the following from the MONITOR output. If the application is invoked by different users from a different

user account, then you can put all the users into a class to limit the amount of CPU the processes can consume.



Class scheduler database file

SYSMAN creates a class scheduler database file called VMS$CLASS_SCHEDULE.DATA when the first scheduling class is created by the CLASS_SCHEDULE ADD SYSMAN command. This database resides on the system disk in SYS$SYSTEM:VMS$CLASS_SCHEDULE.DATA and is an RMS-indexed file. This database file is a permanent database file, which allows OpenVMS to class schedule processes automatically after a system has been booted and rebooted.

In case you want to have a common class scheduler database on a cluster with different system disks, or if you want to have a separate database for each node on a cluster with a common system disk, then on each node, you must define the system logical name, VMS$CLASS_SCHEDULE to point to the location of the database that you want that node to use.
For example:

```
$ DEFINE/SYSTEM VMS$CLASS_SCHEDULE
disk:[directory]VMS$CLASS_SCHEDULE.DATA
```

Process creation

Once a scheduling class is added, the SYSUAF file is updated. By using the class scheduler, a process is placed into a scheduling class at process creation time. Once a new process is created, it needs to be determined whether this process belongs to a scheduling class based upon the data present in the SYSUAF file. The Loginout image obtains the process information from SYSUAF file. Loginout class schedules the process if it determines that the process belongs to a scheduling class. Adding the scheduling class does not affect a process already running.

There are two types of processes: subprocess and detached process. A subprocess becomes part of the same scheduling class as the parent process, even though it may not match the class's criteria. That is, its user and account name and/or UIC may not be part of the class's record. A detached process only joins a scheduling class if it executes the Loginout image during process creation.

Though a process can join a scheduling class at process creation time, user can change or modify its scheduling class during runtime with the $ SET PROCESS/SCHEDULING_CLASS command.

How to determine if a process is class scheduled?

To determine whether a process is class scheduled, use one of the following methods:

SHOW command

You can find out if a process is class scheduled by issuing the $ SHOW PROCESS/SCHEDULING_CLASS command. You can also use $ SHOW SYSTEM /SCHEDULING_CLASS[=class_name], which displays processes that belong to a specific scheduling class (class_name). If the class name is not specified, all class scheduled processes are displayed along with the name of their scheduling class.

SYS$GETJPI system service

The JPI$_CLASS_NAME item code in the system service $GETJPI can be used to get the name of the scheduling class the process belongs to. If the process is not class scheduled, then this system service returns zero (0) to the caller.

Authorize utility

When a new user is added to the SYSUAF file, or when a user's record is modified, the Authorize utility searches the class scheduler database file to determine if this user is a member of a scheduling class. If the user is a member, then Authorize displays "UAF-I-SCHEDCLASS", which indicates that the user is a member of a scheduling class.

SYS$SCHED system service

The SYS$SCHED system service allows you to create a temporary class scheduling database. The processes are class-scheduled by PID, after the process has been created. The SYSMAN interface creates a separate and permanent class scheduling database that schedules you at process creation time. A process cannot belong to both databases, the SYS$SCHED and SYSMAN database. Therefore, the SYS$SCHED system service checks to see if the process to be inserted into a scheduling class is already class scheduled before it attempts to place the specified process into a scheduling class. If it is already class scheduled, then the SS$_INSCHEDCLASS error message, is returned from SYS$SCHED. There is an example of this in sys$examples:class.c on a running OpenVMS system.

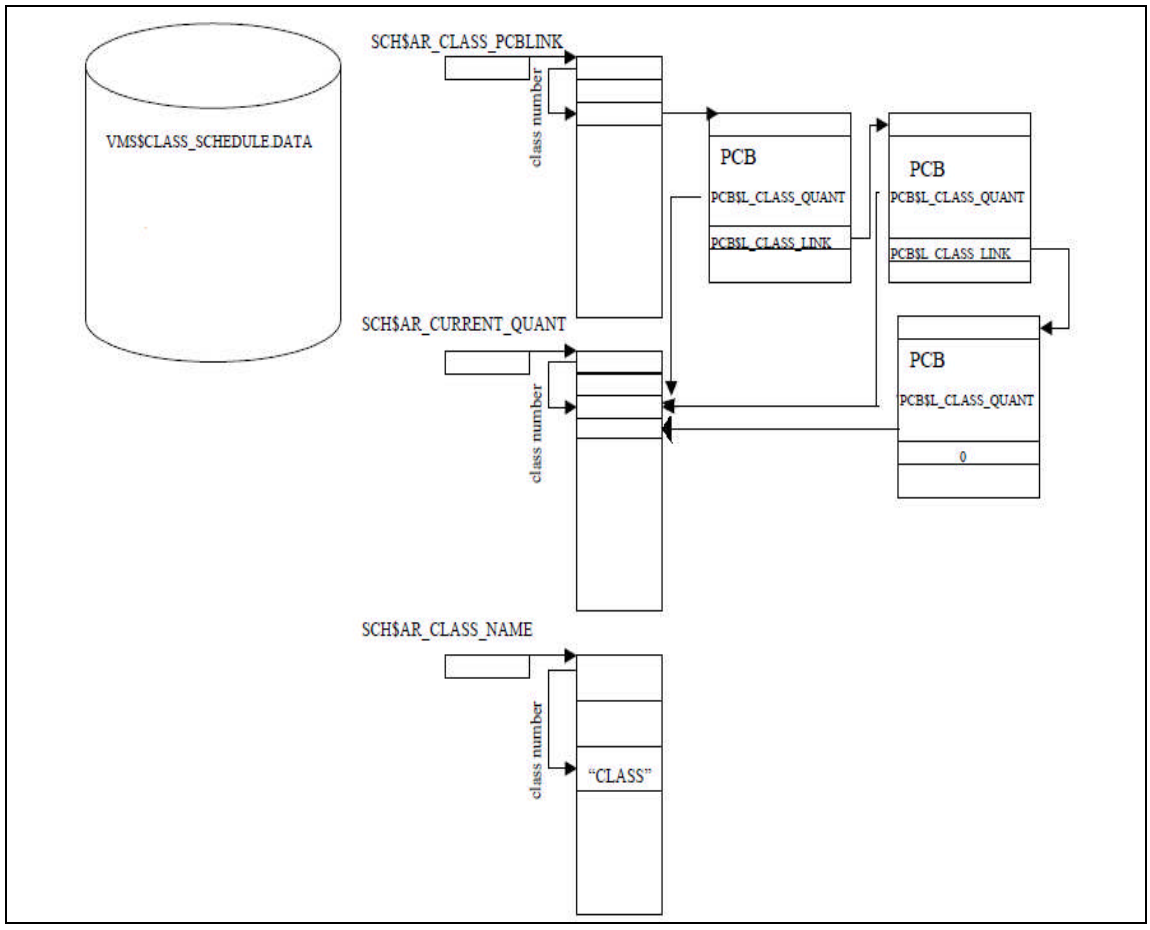
In-memory data structure

The CLASS_SCHEDULE ADD command adds the class to the scheduler database file and creates the in-memory database array, that is SCH$AR_ORIGINAL_QUANT. When you log in, if this in-memory database array exists then, Loginout image calls the SYSMAN routine to find the scheduling class associated with that account. If the logged in user's username or UIC or account belongs to the scheduling class, then SYSMAN routine puts the process which belongs to the account into the scheduling class by updating the below PCB fields of the process:

- Bit PCB$V_CLASS_SCHEDULED indicates that this process is subject to class scheduling. This bit is part of PCB$L_STS2.
- Bit PCB$V_CLASS_SUPPLIED indicates that a class scheduler has specified a class for this process. This bit is also part of PCB$L_STS2.
- Process PCB is placed in a linked list of PCBs for all members of the class. The list is located by indexing, using the class number, into an array pointed to by SCH$AR_CLASS_PCBLINK. The field PCB$L_CLASS_LINK points to the next PCB in this class, if any. Bit

PCB$V_WINDFALL indicates that the process is eligible for scheduling, even though its class is out of quantum, if there are no other eligible kernel threads to schedule.

- PCB$L_CLASS_QUANT contains the address of a class quantum data structure (that is SCH$AR_CURRENT_QUANT) created by the $SCHED system service in non-paged pool when a new class is defined.
- Class name is stored in an array of 16 byte entries pointed to by SCH$AR_CLASS_NAME.
- Primary/secondary day/hour restrictions are described in the array SCH$AR_TIME_RESTRICT.



Class Scheduling Database

SDA example

```
$ ANALYZE/SYSTEM

OpenVMS system analyzer

SDA>
SDA> show class     ! Displays information about active
                    ! scheduling class in the system or dump being
                    ! analyzed.

Scheduling Classes
------------------

                  Original  Current     Time    Process
   Class Name     Quantum   Quantum   Restrict   Count
---------------  --------  --------  --------  --------
MAINCLASS         0000001E  0000001E  0003F800  00000001
SDA>
SDA> show process
Process index: 002E   Name: KETAN             Extended PID: 0000042E
------------------------------------------------------------------------
Process status:          02040001  RES,PHDRES,INTER
        status2:         01608000

PCB address              88362EC0    JIB address              88363B40
PHD address              8E8BC000    Swapfile disk address    00000000
KTB vector address       883631E8    HWPCB address   FFFFFFFF.8E8BC080
Callback vector address  00000000    Termination mailbox          0000
Master internal PID      0001002E    Subprocess count                0
Creator extended PID     00000000    Creator internal PID     00000000
Previous CPU Id          00000001    Current CPU Id           00000001
Previous ASNSEQ  0000000000000001    Previous ASN     0000000000FFDFA2
Initial process priority        4    # open files remaining
126/128
Delete pending count            0    Direct I/O count/limit
150/150
UIC                [00200,000006]    Buffered I/O count/limit
150/150
Abs time of last event   001D1A08    BUFIO byte count/limit
127552/127552
# of threads                   1    ASTs remaining
297/300
Swapped copy of LEFC0    00000000    Timer entries remaining
100/100
Swapped copy of LEFC1    00000000    Active page table count         0
Global cluster 2 pointer 00000000    Process WS page count         478

    Press RETURN for more.
SDA>

Process index: 002E   Name: KETAN             Extended PID: 0000042E
------------------------------------------------------------------------
Global cluster 3 pointer 00000000    Global WS page count           57
PCB Specific Spinlock    88363800    Subprocesses in job             0

! scheduling class associated with the process
Scheduling class         "MAINCLASS"
Original Quantum         0000001E    Current Quantum          0000001E

    Press RETURN for more.
SDA>
```

```
Process index: 002E   Name: KETAN            Extended PID: 0000042E
------------------------------------------------------------------------

Thread index: 0000
------------------
Current capabilities:    System:     000C               QUORUM,RUN
                         User:       0000.00000000
Permanent capabilities:  System:     000C               QUORUM,RUN
                         User:       0000.00000000

Current affinities:                  00000000.00000000
Permanent affinities:                00000000.00000000

Thread status:                       02040001

! scheduling class associated with the process
       status2:                      01608000
CLASS_SCHED_PERM,CLASS_SCHEDULED,CLASS_SUPPLIED,WINDFALL

KTB address              88362EC0    HWPCB address   FFFFFFFF.8E8BC080
PKTA address             7FFEFF98    Callback vector address  00000000
Internal PID             0001002E    Callback error           00000000
Extended PID             0000042E    Current CPU id           00000001
State                     CUR 001    Flags                    00000000

    Press RETURN for more.
SDA>
```

Reference documentation

For more information, please referto the following documents:

- OpenVMS System Manager's Manual, Volume 1: Essentials
- OpenVMS Programming Concepts Manual,Volume I
- OpenVMS System Services Reference Manual: GETUTC–Z