Process Class Scheduling
Using $SCHED

DECUS ’94 National Event, New Orleans

VS027

David L. Cathey
Montagar Software Concepts
P. O. Box 260772
Plano, TX 75026-0772
davidc@montagar.com

Session Outline

- Why Use a Class Scheduler?
- Class Scheduling Components
- Enabling Class Scheduling
- New System Service - $SCHED
- How Class Scheduling Works
- Example of a Class Scheduled System

Why Use a Class Scheduler?

- Class Scheduling provides a method to provide a "quota" on the percentage of CPU a process can use.
- Class Scheduling is more flexible than normal SYSUAF quotas. Processes are allocated CPU resources by a local or third part agent that meets local scheduling requirements.
- Class Scheduling provides more control over CPU allocation than process priority - lower priority processes can be assured some minimal amount of CPU.

Why Use a Class Scheduler?

- Class Scheduling can be used to assure that certain processes receive guaranteed amounts of CPU, such as production jobs, while preventing others from hogging the system.

- Provide a more "level response" by giving a range of users similar resources. Changes in load do not overly effect the "feel" of the system by other users.

- (Sneaky) To restrict the amount of CPU to those playing Galactic Trader, so we can get work done... :-)

Class Scheduling Components

- The OpenVMS Scheduler & Working Set Management
  - Provides the process scheduling primary functions and quantum end processing.
  - Provides callouts (hooks) for the Class Scheduler routines.
  - Provides for scheduling processes granted "windfall" CPU resources

Class Scheduling Components

- The CLASS_SCHEDULER Loadable Executive Image
  - Provides the $SCHED System Service
  - Provides callout routines to the Scheduler and WS Management to track class CPU quantum.
  - Provides for disabling Class Scheduling if loss of Class Scheduling agent.
  - Manages Class Scheduling and related process data structures.

Class Scheduling Components

- The Class Scheduler Process or Agent
  - Defines and implements Class Scheduling policy.
  - Creates scheduling classes and places processes in appropriate Scheduling Class.
  - Allocates CPU to Scheduling Class at periodic intervals.
  - Periodically checks the system for new processes, and places them in a Scheduling Class.

Enabling Class Scheduling

- To enable Class Scheduling, the Loadable Executive Image CLASS_SCHEDULER must be loaded.

- The initialization for this image will provide the body of the $SCHED System Service, and vector other OpenVMS callouts to routines that it provides.

- First, use SYSMAN to define the loadable image and enable loading of external images:

```
$ MCR SYSMAN
SYSMAN> SYS_LOAD ADD VMS CLASS_SCHEDULER
SYSMAN> PARAM USE CURRENT
SYSMAN> PARAM SET LOAD_SYS_IMAGES 1
SYSMAN> PARAM WRITE CURRENT
SYSMAN> EXIT
```

Enabling Class Scheduling

- Now, update the data file read by EXEC$INIT that lists the images to be loaded at boot-time:

```
@SYS$UPDATE:VMS$SYSTEM_IMAGES
```

- Reboot the system at your leisure...

New System Service - $SCHED

- The $SCHED System Service is the interface into the Class Scheduler subsystem. It provides the following services:
  - — Enables Class Scheduling to occur.
  - — Assign a process or set of processes to a Scheduling Class.
  - — Allocates CPU quantum to the defined Scheduling Classes.

New System Service - $SCHED

- The $SCHED System Service has the following call interface:

  status = SYS$SCHED(func, p1, p2, p3)

  func - function code, read-only by value

  p1 - parameter 1, usage by function code

  p2 - parameter 2, usage by function code

  p3 - parameter 3, usage by function code

New System Service - $SCHED

- The following function codes are defined:
  - — CSH$_READ_ALL Gets information on all processes
  - — CSH$_READ_NEW Gets information on unseen processes
  - — CSH$_READ_QUANT Gets class quantum values
  - — CSH$_SET_ATTN_AST Set an attention AST if new processes seen
  - — CSH$_SET_CLASS Place processes into scheduling classes
  - — CSH$_SET_NEW Make a process unseen (Kernel/Exec only)
  - — CSH$_SET_QUANT Give classes CPU allocation
  - — CSH$_SET_TIMEOUT Set timeout for disabling Class Scheduler

How Class Scheduling Works

- Although more of the features can be used, a simple Class Scheduler Process can be implemented using the following basic functions:

  — CSH$_READ_NEW - To find processes that have not yet been class scheduled (initially, all of them).

  — CSH$_SET_QUANT - To allocate CPU quantum to each Scheduling Class at periodic intervales.

  — CSH$_SET_CLASS - To place selected processes into a Scheduling Class.

How Class Scheduling Works

- CSH$_READ_NEW - Scans all processes on the system and will fill in an array of CSHP structures. Each structure member provides the following information about a process:

  — The Process Identification

  — A Process Index

  — Current Priority

  — Base Priority

  — Account field, typically from SYSUAF value

  — Accumulated CPU time (in 10-millisecond tics)

How Class Scheduling Works

- Depending upon the Class Scheduling policies that would be desired, additional information about the process may be required (batch, privileges, image, etc).

How Class Scheduling Works

- Example call using CSH$_READ_NEW:

```
#include   <cshdef.h>
CSHP       proc_info[200] ;
/*
 * struct cshp$ {
 *          int        cshp$l_status ;
 *          short      cshp$w_pix ;
 *          char       cshp$b_pri ;
 *          char       cshp$b_prib ;
 *          int        cshp$l_epid ;
 *          char       cshp$t_account[CSHP$S_ACCOUNT] ;
 *          int        cshp$l_cputim ;
 * } ;
 */
int        proc_info_size ;
int        proc_info_retlen ;
proc_info_size = sizeof(CSHP$K_LENGTH) * 200 ;
status =  SYS$SCHED(CSH$_READ_NEW, proc_info, &proc_info_size,
           proc_info_retlen) ;
```

How Class Scheduling Works

- CSH$_SET_QUANT - Grants Scheduling Classes amounts of CPU in 10-millisecond tics.
- A Scheduling Class **must** have quantum provided to it at least once before a process can be placed into that Scheduling Class.
- Scheduling Classes are numeric values from 0-8191.
- CSH$_SET_QUANT simply requires an array of longwords, each longword indicating the amount of CPU to be given to that Scheduling Class index.

How Class Scheduling Works

- A CSH$_SET_QUANT must be done at periodic intervals (default 30 seconds) or Class Scheduling will disable itself to prevent processes from permantently stalling.

- Example of call using CSH$_SET_QUANT:

```
int         class_quantum[8192]
for( i = 0 ; i < max_class ; i++ )
            class_quantum[i] = determine_ticks(i) ;
class_size = 4 * max_class ;
status = SYS$SCHED(CSH$_SET_QUANT, class_quantum, &class_size, 0) ;
```

How Class Scheduling Works

- CSH$_SET_CLASS - Places processes into Scheduling Classes.   Which Scheduling Class is dependent upon the local policy of assigning Scheduling Classes and the amount of CPU quantum each Class is given at periodic intervals.

- It may be desirable to allow some processes to exceed the amount of quantum for a Scheduling Class if the CPU would other wise be idle. These processes should be given *windfall* CPU resources.

How Class Scheduling Works

- $SCHED is given an array of CSHC structures indicating which processes are to be Class Scheduled. For each processes, the following must be provided:
  - — Process Identification
  - — Scheduling Class Number (0 - 8191, -1)
  - — Windfall CPU Flag

How Class Scheduling Works

- Example call using CSH$_SET_CLASS

```
#include   <cshdef.h>
CSHC       proc_class[200] ;
/*
 * struct cshc$ {
 *          int         cshc$l_epid ;
 *          short       cshc$w_class ;
 *          short       cshc$w_windfall ;
 * } ;
 */

for( i = 0  ; i < (proc_info_retlen/CSHP$K_LENGTH) ; i++ ) {
            proc_class[i].cshc$l_epid = proc_info[i].cshp$l_epid ;
            proc_class[i].cshc$w_class = determine_class(&proc_info[i]) ;
            proc_class[i].cshc$w_windfall = determine_windfall(&proc_info[i]) ;
}
class_size = i * CSHC$K_LENGTH ;
status = SYS$SCHED(CSH$_SET_CLASS, proc_class, &cshc_size,
             &CSHC$K_LENGTH) ;
```

Example of a Class Scheduled System

- It is desired to provide for a consistent "response-time" for user processes, even with variances in system load. Background type processes should be able to absorb idle CPU.
- A Class Scheduler Process is developed to provide 10% of the CPU to each interactive process. All other processes are placed in a "batch" Scheduling Class with some CPU, but granted windfall CPU.

Example of a Class Scheduled System

- The Class Scheduler Process will allocate CPU at 1 second intervals to provide a smoother feel if processes consume their entire class quantum.
- Three test processes (infinite loops) are launched to see how the system will react...

Example of a Class Scheduled System

- Sample output of showing Scheduling Class usage:

```
                    OpenVMS Class Scheduler Utilitization
                               on node AVATAR
                            4-MAY-1994 13:26:32

Scheduling Class Name           0         25        50        75       100
                                + - - - - + - - - - + - - - - + - - - - -+
DEFAULT                    28   XXXXXXXXXXX
                                |         |         |         |          |
DAVIDC                     10   XXXX
                                |         |         |         |          |
JOHNW                      10   XXXX
                                |         |         |         |          |
RICKR                      10   XXXX
                                |         |         |         |          |
<Unscheduled>               5   XX
                                |         |         |         |          |
```

Example of a Class Scheduled System

- Note that idle CPU is still available. Each job has the same "feel", even if the load changes (assuming some amount of idle CPU).
- The interactive jobs were infinite loops, yet were not able to consume the entire CPU. Other jobs were given CPU as needed.
- Process priorities effectiveness only provide a strong effect within a Scheduling Class. If no quantum is available, even a higher priority process must wait to have Class quantum refreshed.

Additional Reading

- OpenVMS System Services Reference Manual
- An example program is provided in:

  SYS$EXAMPLES:CLASS.C

- For those that enjoy this sort of thing, the OpenVMS Source Listings for V6.0:

  Module SYSSCHED.LIS in [SYS.SRC]

  Module SCHED in [SYS.SRC]

  Module RSE in [SYS.SRC]

Questions and Answers

David L. Cathey
Montagar Software Concepts
P.O.Box 260772
Plano, TX 75026-0772

(214) 618-2117