

MPE Queuing: How It Works, How To Make It Work For You!

One of the very key concepts to understand when approaching MPE performance is that of the queuing algorithm used to decide which process gets to execute on the CPU. It seems that this concept, although once caught seems easy, can be difficult to catch. I have had numerous requests to try and explain it in written format as I teach performance classes. So here goes! I hope you are able to catch it!

What is Queuing?

MPE/iX systems are queuing systems. A queue is defined, in computer terms, as “a sequence of stored data or programs awaiting processing.” In the greater sense there are a number of different queues involved in computer performance. This means that the computer is really a system of queues. Queues are places where requests for the computers time must wait for the attention of a resource. Queues allow for the co-existence of many applications and their processes in the Multi-Processing Environment (MPE). Queues are a scheme which allow many processes to be executed in a “concurrent” manner. The process is the basic unit of activity in a computer. Every request for CPU time must have a process and that process must be assigned a priority.

In discussing the MPE Queuing Algorithm we are talking about the queue that controls the activity of the CPU. This is often referred to as the Ready Queue. The process that determines the nature of any given process and what priority it deserves within the queue is called the Dispatcher.

How does the Dispatcher Work?

Understanding the method the Dispatcher uses in assigning and adjusting priorities is the key concept in understanding system queuing and thus system performance. The Dispatcher is the entity that assigns a process' priority, gives the process time on the CPU (this is called Launching), monitors the process' life on the system, adjusts the process' priority (usually down in priority), and sees that the process gets the CPU again (when it becomes ready).

What are the rules that the Dispatcher uses for controlling queues?

In assigning priorities the Dispatcher uses a numeric range from 0 to 255. These numeric values are assigned to five basic scheduling or execution queues. There is a bit more complexity to the whole thing than this, chiefly because of the addition of the Workload Manager product (at release 5.0 this expanded the potential queues from the basic abilities found in the “classic” queuing we have all become comfortable with). In the figure that follows the very bottom portion of the SHOWQ command (or SHOWQ ;STATUS) shows the queues and their setup:

-----QUANTUM-----							
QUEUE	BASE	LIMIT	MIN	MAX	ACTUAL	BOOST	TIMESLICE
CQ	152	200	1	2000	114	DECAY	200
DQ	202	238	2000	2000	2000	DECAY	200
EQ	240	253	2000	2000	2000	DECAY	200

Showq output

The SHOWQ command actually shows only the three queues that are generally used for most user activity. There are in reality five MPE execution queues.

In the figure below the queues are shown using a screen found in SOS/3000 from Lund Performance Solutions (LPS). The execution queues are found in the third section under the heading "Scheduling Information".

```

SOS/3000 E.11v(c) LPS  TUE, APR 29, 1997, 11:15 AM  E: 00:25:54  I: 01:08
----- System Configuration -----
|CPU Type   : 947LX      HPSUSAN: 742502121    MPE/iX Version: B.40.00 |
|Memory Size: 160M     Physical Console: 20  OS AIF Version: A.03.01 |
|User Mode  : MULTI    Logical Console : 20  MI AIF Version: A.02.00 |
----- Job and Session Information -----
|Jobfence   : 7        Job Limit  : 22      Job Count  : 14    Next Job # : 43 |
|Outfence   : 7        Sess Limit : 99      Sess Count : 74    Next Sess #: 106 |
|Jobsecurity: HIGH     Streams Dev: 10      |
|Max # J/S  : 2500    Max # Procs: 5460    Max # Open Files/Process: 1024 |
----- Scheduling Information -----
| Queue  Base  Limit  Quantum  Maximum  Minimum  Time Slice  Boost |
| AS     30   99      |
| BS     100  150      |
| CS     152  200      49        2000     1         200        DECAF |
| DS     202  238      2000      |
| ES     240  253      2000      2000     2000     2000        DECAF |
-----
Enter Command:

```

Sos/3000 System Configuration Screen

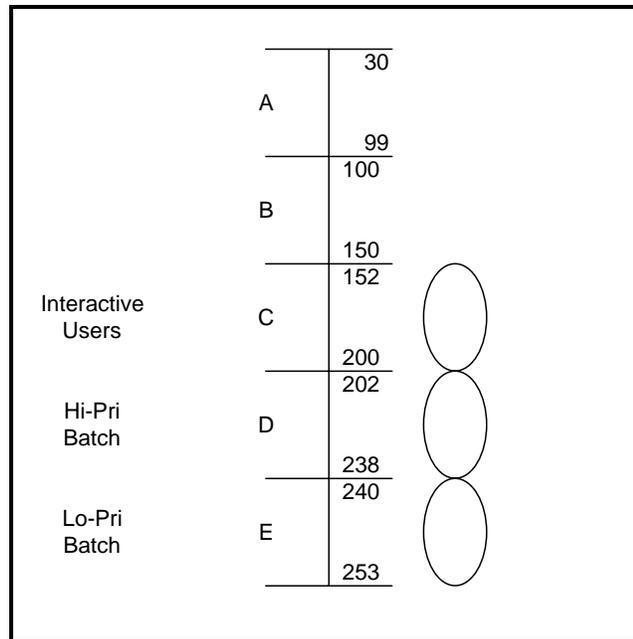
In both the output from the ShowQ and the System Configuration screen from within SOS/3000 there are several columns. It is important to have an understanding of those columns and what they mean.

- Queue - This is the queue name. Sometimes "CQ" is used to refer to the queue. Sometimes "CS" is used. When an "S" is used next to the queue reference it often means that the queue is circular in nature (that is that the queue priority values change during the life of the process).
- Base - This is the highest priority value in the given queue (or top) .
- Limit - This the lowest priority value in the queue (or bottom) .

MPE Queuing: How It Works, How To Make It Work For You!

- Quantum (the same as Actual on the SHOWQ) - This the value that the Dispatcher has calculated for the queue. It represents the average amount of time the normal transactions use. This is measured in milliseconds and is calculated dynamically by the Dispatcher. By default the C queue's Quantum is calculated continually by the Dispatcher as it monitors that queue. Historically the D and E queues had their values set to 2000 milliseconds. A Quantum is the value the Dispatcher assigns to a process before the Dispatcher again adjusts the processes priority and compares it's priority to that of others requesting the CPU.
- Maximum (Max) -This is the maximum value that a quantum value can be limited to.
- Minimum (Min) - The minimum value that a quantum that can be limited to.
- Time Slice - This is the length of time a process is given before it generates an interrupt. In order to avoid having a CPU bond process take up all of the CPU time an interrupt is generated. This allows for a check of the processes status and is intended to keep that process from taking up all of the available CPU time.
- Boost - This value controls what happens to a process when it reaches the bottom of a given queue. The word "Decay" here means that the process retains the value at the bottom of the queue. The word "Oscillate" means that the Dispatcher will bounce the processes priority up to the top value of the given queue when the bottom is reached.

When the queuing algorithm is at work it handles things in the fashion described in the picture that follows. This is the usual default setup that I have found in place in about 90 percent of all installations (sometimes it is left that way because the repercussions of making changes to it were not understood).



Queuing Illustration

In the Queuing illustration there are five basic queues. These are given a range of priority queues that start with 30 and end with 253 (along the continuum that was mentioned earlier from 0 to 255). The five queues allow the Dispatcher to assign a queue and thus a numeric priority to processes in such a way that those of higher importance get higher priority. This is how the various queues work:

- The A queue is reserved for very important system processes. These processes absolutely must have the CPU immediately. The A queue has a range of 30 to 99. Processes of a lower priority will lose the CPU to processes in this queue (this is called Preemption).
- The B queue is usually reserved for important system processes also. However, this can be used for some carefully placed user applications. Performance monitors typically run here since all of their information must be gathered at the same time. But other applications can be carefully placed in this queue for better performance. The B queue has a range of 100 to 150.
- The C queue is typically used for interactive users. This queue typically has a range of 152 to 200.
- The D queue is typically used for high priority batch processing. This queue typically has a range of 202 to 238.

- The E queue is typically used for low priority batch processing. This queue typically has a range of 240 to 253.

The SHOWQ command shows the execution priority but nothing more, so I have used the statistics from the global screen within SOS/3000 to help explain more about processes and their priorities.

PIN	J/S#	Session/User Name	Cmd/Program	CPU%	QPri	#Rd	#Wr	LDV	#Tr	PRes
4	<sys>	<system process>		.<	AL13	0	0	-	0	-
87	<sys>	<system process>	JOB	.<	BL100	0	0	-	0	-
6	<sys>	<system process>		.<	CL152	0	0	-	0	-
36	J18	FAX,RAFX.H3000	RAFX	.<	DS202	0	0	10	0	-
24	<sys>	<system process>	NMCONSOL	.<	BL149	0	0	-	0	-
308	J41	DKPURGE,PURGE.H3000	RAPG1	.1	DS238	2	0	10	0	-
7	<sys>	<system process>		.1	CL152	0	0	-	0	-
15	<sys>	<system process>		.1	AL13	0	0	-	0	-
119	J10	PRINT,RAPB.H3000	RAPB	.1	CS152	0	0	10	0	-
23	<sys>	<system process>	NMTRCMON	.2	BL149	0	0	-	0	-
2	<sys>	<system process>	LOAD	.3	BL142	0	0	-	0	-
329	<sys>	<system process>	VTSERVER	.3	CS152	0	0	-	0	-
5	<sys>	<system process>		.3	CL152	0	11	-	0	-
125	<sys>	<system process>		.4	BL146	0	0	-	0	-
3	<sys>	<system process>		.7	BL100	0	0	-	0	-
260	S9	FILE.H3000	RASYPO	1.2	CS152	0	0	125	0	-
179	<sys>	<system process>	VTSERVER	1.5	CS152	0	0	-	0	-
332	S17	P029SKSR,TECH.H3000	RASYPO	1.5	-----	3	0	26	2	.2
182	S20	TECH.H3000	RASYPV	1.6	CS152	0	0	102	3	.1
178	S17	P029SKSR,TECH.H3000	RASYPV	1.9	CS152	0	0	26	1	.5
190	J31	OUTGOING,DSP4.H3000	RIDS4	2.6	DS202	0	2	10	0	-
232	S39	TECH.H3000	RASYPV	3.0	CS152	1	0	113	6	.1
290	J39	RECEIVE,DRP4.H3000	RIDR4	3.1	DS204	2	1	10	0	-
403	S77	TECH.H3000	RASYPO	3.6	CS152	4	0	210	4	.4
107	J12	VERIFY,DV1.H3000	RIDV1	5.2	DS202	2	2	10	0	-
365	S102	MGR.LPS	SOS	6.4	BS100	8	0	603	1	.<
193	J32	OUTGOING,DV2.H3000	RIDV2	6.6	DS202	0	1	10	0	-
311	J42	DKQPURGE,BATCH.H3000	RIPG1	7.1	DS238	10	2	10	0	-
285	S101	STEVE,MGR.LPS	SOS	7.7	BS100	1	0	64	0	-
448	S17	P029SKSR,TECH.H3000	RASYPO	8.7	CS152	0	0	26	0	-
184	J35	TECHPROD,MGR.H3000	UDMSSHR	29.6	DS238	6	0	10	0	-

SOS/3000 Process Detail output

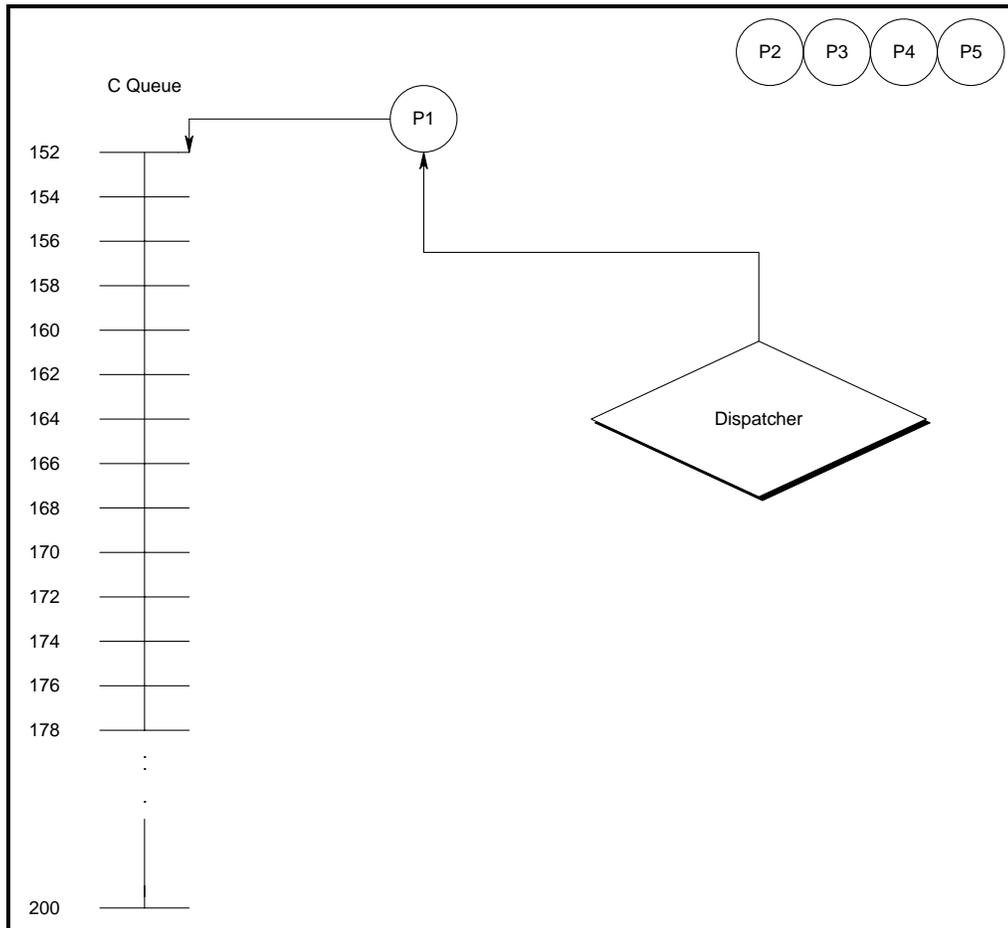
The SOS/3000 Process statistics output shows a number of headings. However the ones we are concerned with are the PIN – Pin Number (every process must be assigned a pin, the Job or Session number, the Session/User Name, the CMD/PROGRAM name and the QPRI. The QPRI column consists of a four or five character value. The first letter is the priority (A - E), the second is either an S or an L (L meaning linear - this kind does not have it's priority changed by the Dispatcher, S meaning circular – this kind can be adjusted by the Dispatcher). The third through fifth values are the numeric priorities assigned by the Dispatcher. While watching a process via a monitor such as SOS/3000 the priorities of those processes with C, D, or E priorities should be adjusted downward (unless the Oscillate option for decay has been turned on for that queue).

Why does the Dispatcher adjust priorities?

MPE Queuing: How It Works, How To Make It Work For You!

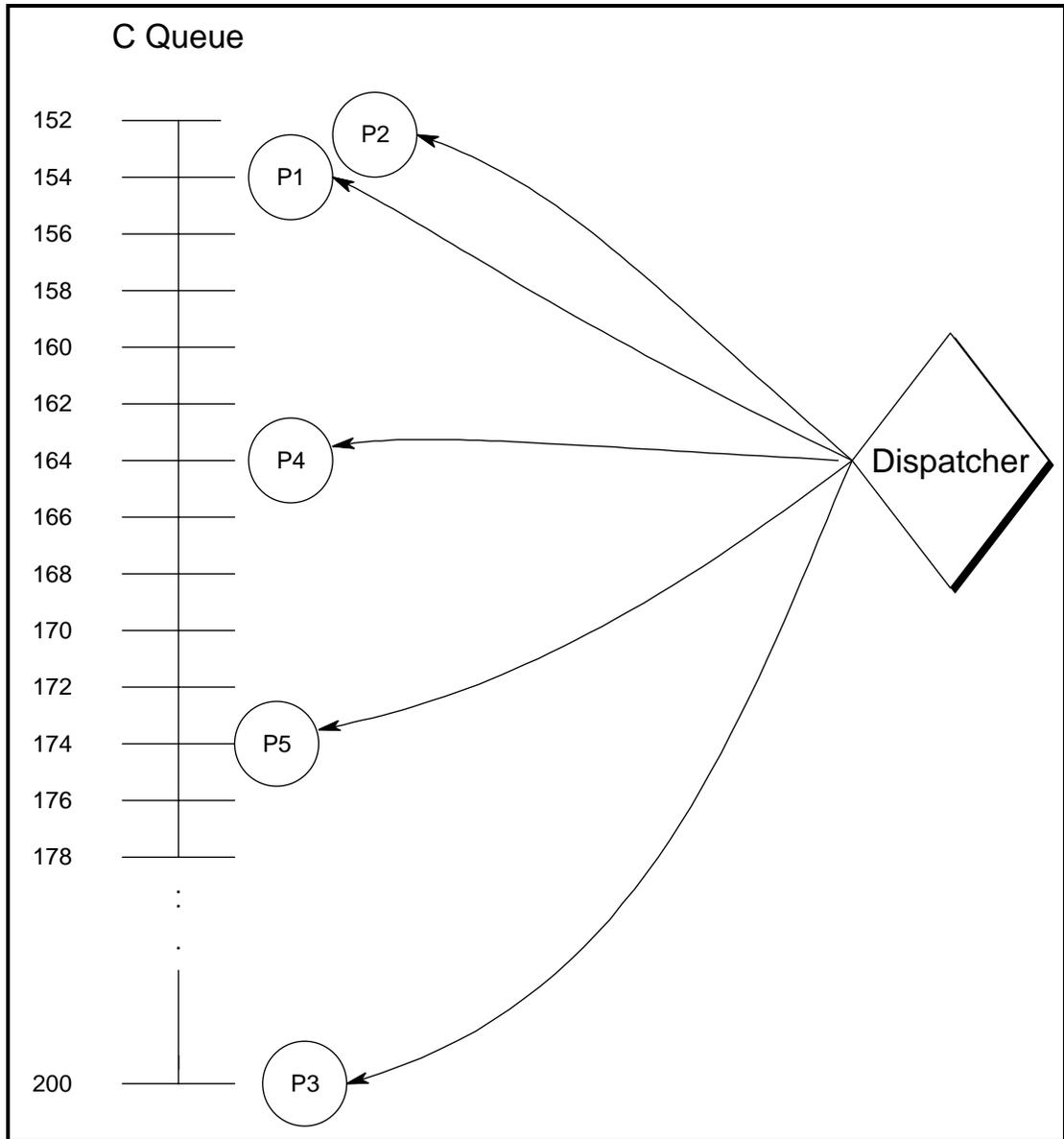
The Dispatcher monitors and adjusts the priority of a process (usually by increasing the value – thus lowering it's ability to compete for the CPU). This is called Dispatcher Decay. Why is the Dispatcher designed to do this? Because the design of the algorithm is intended to help processes that use the CPU for a very short amount of time get completed. The design, however, will penalize those processes who use it for a very long time by dropping their priority. This is called PSPTF or "Preemptive Shortest Processing Time First". This is done to keep more lengthy transactions from hogging the CPU and keeping others from getting CPU time.

The roll of the Dispatcher looks like the illustration below. As processes begin making a demand on the CPU only one process is launched on the CPU. The figure below uses just the C queue in the illustration. The circles with the P1 through P5 represent the individual processes. Only one process has requested CPU time. That process will get all the CPU time it needs until it blocks for another resource or is completed.



C Queue Illustration

In the next figure many more processes are competing for CPU time. In this example the Dispatcher must work much harder. It must keep track of each processes priority, dynamically calculate the quantum, preempt those which because of Dispatcher Decay no longer deserve the CPU and launch processes on the CPU (those processes which fall below the numeric and alphabetic priority of others have the potential to be PRE-EMPTED in their use of the CPU). This work takes CPU time so busier systems will begin to notice higher and higher amounts of CPU time allocated to the Dispatcher.



C Queue Illustration

What is Process Preemption?

Process Preemption means that a processes priority has dropped below that of another process with a higher priority (that is also ready for execution - which means it has it's code and data ready). Process Preemption can be measured as what is called a WAIT STATE. On very busy systems the percentage of this Preemption Wait State can become very high. The Dispatcher must work very hard to keep up with this activity and there is CPU assigned to the Dispatcher as a process. The result of Preemption is that individual processes will begin to labor and complete more slowly as a very high percentage of their time is spent being Pre-empted.

The Process and Preemption

This is how the process works. The Dispatcher assigns a priority to each process in the Ready Queue. The highest priority process that is ready is launched on the CPU. If that process continues processing without needing another resource or without another process of a higher priority requesting the CPU it continues until the length of the quantum for it's assigned queue has passed. The Dispatcher will drop the priority, usually in increments of two, and the process will continue. At anytime during the life of the process it can be pre-empted by a higher priority process. It can also be stopped for other resources. If a process is stopped the Dispatcher retains information about the process and how much of the processes time is yet to be used at that priority level. When that process is the highest priority process again and is launched the process continues from where it left off and uses the remainder of it's quantum.

How can you use a proper understanding of MPE Queuing Theory to your benefit?

Once the theory of MPE Queuing is properly understood some small changes to the queuing parameters can help the performance on individual processes or groups of processes. Changes that are made to the default setup should be carefully monitored (remember that changing the queuing parameters is just reallocating the same pie of CPU usage). The default queuing setup has proven to be the best setup in probably 90 percent of the HP3000 sites (by default I mean C queue set to 152 - 200, D queue set to 202 - 238, and E queue set to 240 - 253) so change it with wisdom and care.

What can be changed and how can it help Performance?

In the default settings each queue is discreet, there are no overlaps, a B queue process will always have a higher priority than a C queue, a C queue process will always have a higher priority than a D queue and so on. In this setup processes in the lower priorities only get the CPU when no process of a higher priority is present and ready. One change to the queuing setup that may help with the performance of these lower priority processes (these are often batch jobs) is to change the queues so there is some

MPE Queuing: How It Works, How To Make It Work For You!

overlap. How does this help the performance of these lower priority processes? They can now compete with those processes at the bottom of the queue above them for the CPU. This can enable them to get more CPU time.

Another possibility for queue parameter changes is to use the often unused E queue as a secondary batch queue. This queue can be modified so that it crosses into the C queue and reaches down into the D queue or below. This change would help those jobs placed in the newly defined E queue get a higher percentage of the CPU. Sessions placed here would get an equal opportunity for the CPU's attention until they dropped below the C queue's bottom (or limit) and then would be treated with less preference down in the D and E queue. The figure below shows the potential setup of this newly defined E queue.

```
showq;status
```

QUEUE	BASE	LIMIT	-----QUANTUM-----			BOOST	TIMESLICE
			MIN	MAX	ACTUAL		
CQ	152	200	1	2000	12	DECAY	200
DQ	190	238	2000	2000	2000	DECAY	200
EQ	152	253	2000	2000	2000	DECAY	200

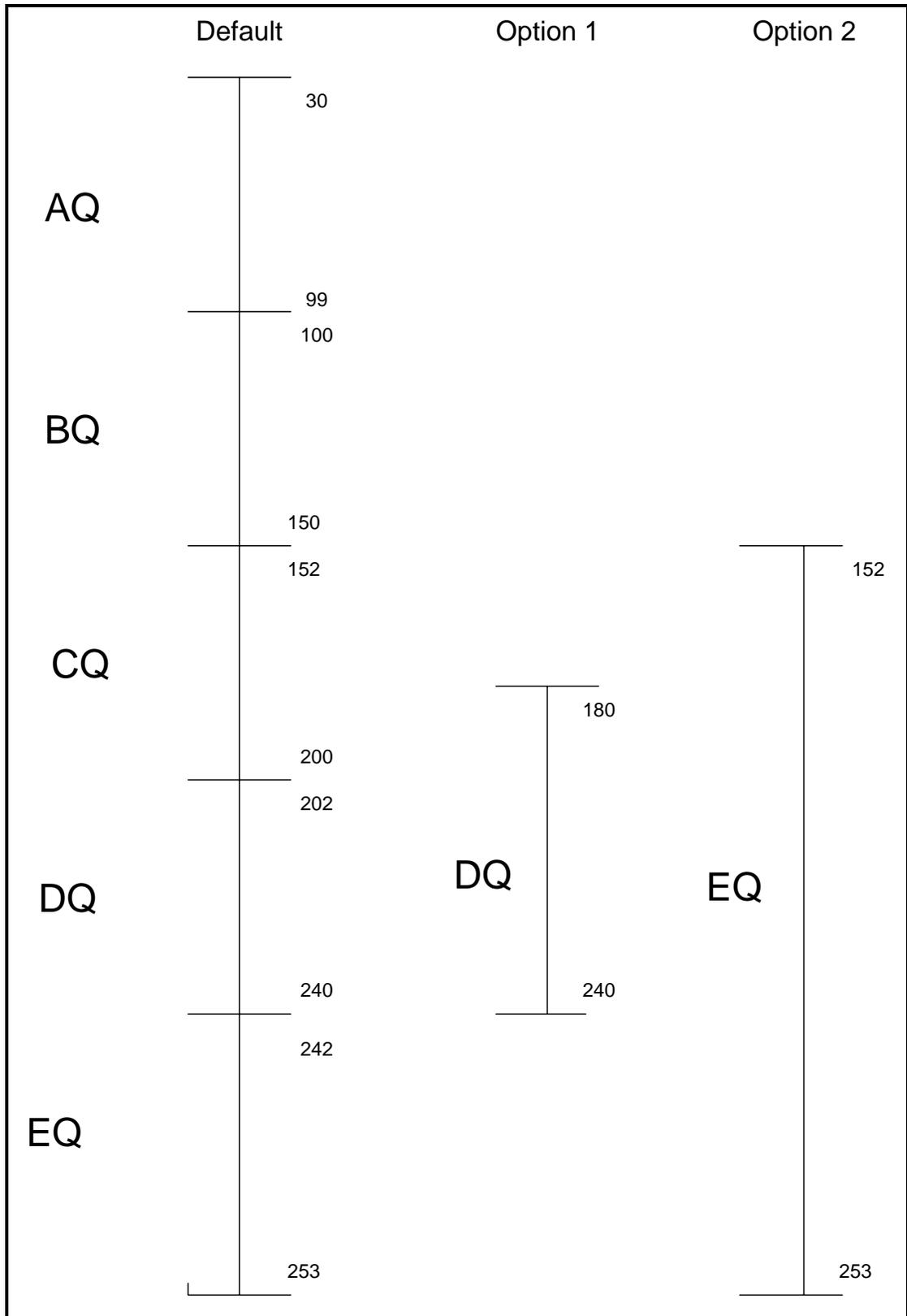
SHOWQ ;Status

Why would one of these strategies be used? The strategy illustrated in Option 1 of the figure that follows (entitled 'Queuing Change Options') would help job throughput and completion. Jobs that took a very short amount of CPU time would compete with sessions that require more CPU time and thus are found near the bottom (or limit) of the queue. This would help the jobs complete more quickly. There are some environments that have an important job or two that must complete in a timely fashion so this strategy would help those jobs.

Jobs placed in the E queue under Option 2 are also afforded a better opportunity to run on the CPU since they compete with processes in the C queue. However, after initially competing with C queue priority processes placed in this redefined queue will end up down in the D queue. Finally, if they are very lengthy jobs they end up down in the E queue and receive whatever CPU is available. Processes placed in this extended and redefined queue are allowed to compete with the on-line users queue for a short time. If they complete during this time they never leave the higher priority range. If they take longer to process they end up in the D queue and then the E queue.

The key understanding about these various options is that under the normal scheme each execution queue gets only the CPU that is left after the higher queue receives what it needs. In the changes we have described the lines between the queues are no longer there and queues will begin to compete with each other for CPU time.

MPE Queuing: How It Works, How To Make It Work For You!



Queuing Change Options

MPE Queuing: How It Works, How To Make It Work For You!

How are the queues changed?

Queues can be modified in several ways. The most common way is by using the TUNE command. The Tune command can change a number of parameters that affect the queuing algorithm.

One is the base (or the top of the queue), and another is the limit (the lowest value in the queue). Adjusting the base and limit can affect performance of processes in a queue by allowing them to run in higher priority which may overlap that of another.

A second thing that can be changed with the TUNE command is the minimum and maximum quantum. This has usually only applied to the C queue but now, with the latest operating system versions it can also work with the D and E queue. What this controls is the calculation of the quantum. If a minimum and maximum value is set to some value higher than zero the value must stay above that level. This would keep the dispatcher from dynamically calculating the value as it normally does unless the calculation was above the minimum threshold. If the calculated value would have been below the threshold, the value is set to the specified minimum. What is the impact of this? Those short transactions that would have completed in less than a quantum are not affected. They cannot be forced to use more CPU than they need. Those more lengthy processes would decrease in priority as they normally would but they would potentially get more CPU time at each priority level since the quantum is longer. This strategy is an effort to help processes that take more time get more CPU before they end up at the bottom of the queue. It helps them get their required CPU before reaching the bottom or limit of the queue.

A third change to consider is switching between the normal setting of Decay and the option of Oscillate. The Decay setting will cause processes in the queue to drop to the bottom (or limit) of the queue and stay there. Oscillate causes processes that reach the bottom of the queue to jump back up to the top. This change helps longer lasting processes compete for the CPU more by jumping them back up to the top of the queue.

Each of these queue parameter changes involves helping the longer and slower processes get more CPU. The default priority scheme does a very good job of reserving the CPU for the very short transactions while preempting those that take longer to complete.

The figure that follows shows the TUNE command parameters and how to use them. To change the tuning of the E queue to range from 152 to 253, for example, you would enter this command:

```
TUNE ;EQ=152,253
```

TUNE

Changes the scheduling characteristics of the scheduling queues. These characteristics include base and limit priorities, quantum bounds (min and max), boost property and timeslice. (NM)

SYNTAX

```
                {CQ}
TUNE[minclockcycle][[:]{DQ}=[base],[[limit][,[min][,[max]
                {EQ}
[,[{decay  }]][,[tslice]]]]]
    {oscillate}

[[:]...]
```

CAUTION

Misuse of this command can significantly degrade system operating efficiency.

TUNE Command Parameters

Priority control for individual processes

Priority control for individual programs, jobstreams or sessions can be done in a number of ways. One way is to place a jobstream into specific queue is by using the PRI=queue statement on the jobs first line. At logon the user can add PRI=queue (BS) which can also be specified as an option logon in a UDC file. The program itself can be placed into a specific queue using a programmatic call.

ALTPROC

Changes characteristics for the specified processes. Currently, you may change the priority, queue attribute and workgroup for a process. This command requires OP or SM capability.

SYNTAX

```
ALTPROC [ [PIN=]{pinspec          }
          {(pinspec [,pinspec ]...)}
          [ [;JOB=]{jobspec          }
            {(jobspec [,jobspec]...)}

          { [;PRI=] pri
            [;WG= ]{workgrp
              NATURAL_WG } }

          [;TREE | ;NOTREE]
          [;USER | ;ANYUSER]

          [;SYSTEM]
```

ALTPROC Command Parameters

Individual processes can be altered to higher or lower priorities using the ALTPROC command. This command requires SM (system manager) or OP capability and can alter by pin, or job number/session number. This works well on a limited basis chiefly because this kind of manipulation requires a lot of time. Changes made in this way last only until the end of the process.

Using the system available commands to modify the definition of the queues can sometimes give only a limited benefit. In more difficult situations a queue management product can provide the power and complexity to help control the competing processes better. Hewlett-Packard offers the very powerful product called Workload Manager while Lund Performance Solutions offers Q-Xcelerator. KLA - express is also a product that allows for a greater degree of control over the execution queues.

HP and third party packages that allow greater control.

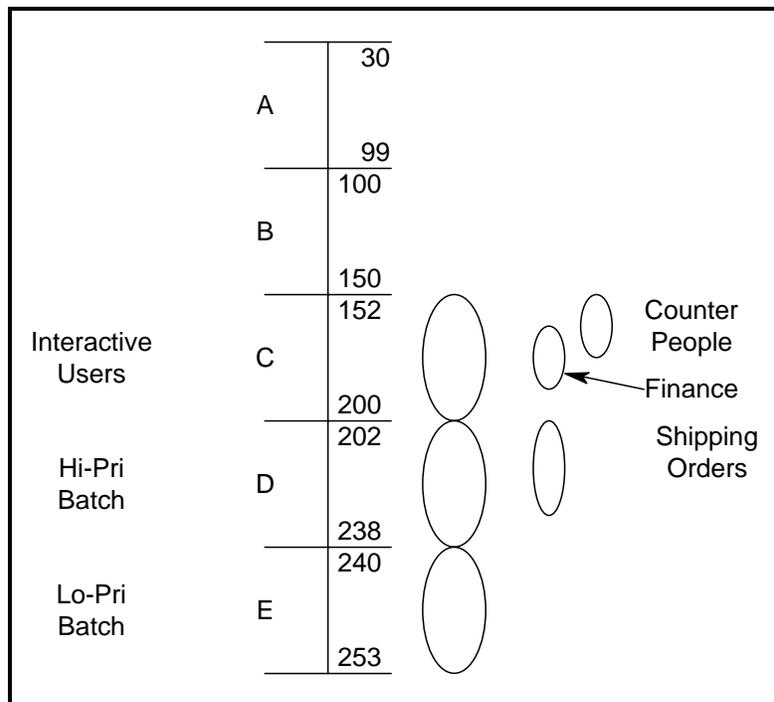
The workload manager product extends the capabilities of the traditional scheduling algorithm. A collection of user or system processes make up a workload. And these workloads allow for more execution queues than the five that exist in the traditional queuing algorithm. A new set of

commands allows for the setup and viewing of the new queues (SHOWWG, ALTWG, NEWWG, PURGEWG, and SHOWWG) and there is some interesting extended capabilities. The new queues can have a minimum and maximum CPU percentage. This means that a guaranteed amount of CPU can be there for a workload regardless of the load on the system.

Queue Management Tools

Queue management tools approach system queue controls from a stand point of controlling the priority. In this scheme controlling the priority is deemed sufficient since only those processes with the highest priority are launched and it is not necessary to maintain a minimum CPU percentage. A product like Q-xcelerator has some other interesting capabilities like the ability to bump from one queue to the next after a certain number of milliseconds have passed. The disadvantage of a queue management tool is that they operate as a job on the system. They take overhead and since they come along after the dispatcher has assigned a process it's queue priority and change the queue they can need to do a lot of work.

The following figure is an example of how a queue management tool can help redistribute the existing execution queues.



Queuing Management setup

What are the benefits of understanding and modifying the queuing parameters?

MPE Queuing: How It Works, How To Make It Work For You!

In perhaps 90 percent of the sites that I have seen nothing has been done to alter the default queuing parameters. In most of these the on-line users are happy with response and the batch jobs complete in reasonable time. However, as a system gets busier and the utilization approaches a point of trouble due to a restriction in CPU resources, modifying the queuing parameters can be a life saver. This step can help extend the useful life of the system (remember that principle that you are robbing Peter to pay Paul). Additionally, in systems where certain batch processes have a higher importance than others, queue modification can save the day. I have helped sites that had a pick list (in an order fulfillment environment) that was absolutely vital and needed a faster response time. But since it normally ran in the B queue it lost out to on-line activity in the higher C queue. Queue management via the ALTPROC command and then via a queue management application helped insure that this program got what it needed.

Queue management can help distribute resources in a better way and allow for distribution of resources peculiar to the specific environment. This will keep users satisfied and management content that the system is being managed well. A good understanding of queue parameters and proper use can maximize the hardware investment, getting more useable life out of the system.

by: Jeff Kubler
Lund Performance Solutions
6/17/97