

**Candle** **Workload Manager and CICS, Revisited**




**Stephen L. Samson**  
Senior Technical Staff Member  
Candle Corporation  
*steve\_samson@candle.com*  
*steve.samson@attglobal.net*  
New York CMG  
November 15, 2002  
New York, New York

Trademarks and registered trademarks used in this presentation are the property of their respective owners and are to be regarded as appearing with the appropriate ™ or ® symbols at their first mention. Predictions and evaluative statements are the opinion of the author only and do not necessarily represent Candle Corporation positions.  
© Copyright: 2000–2002, Candle Corporation. Permission is granted to New York CMG to distribute images of the slides and notes of this presentation to attendees. All other rights reserved.  
SLS 30OCT2002

**CICS Topics: Getting CICS to Play Nicely With Others**

- How WLM manages CICS in goal mode
- New options
- Lessons in humility—living in the real world
- Details of exploiting goal mode for CICS
- Summary



Page 2 NYCMG 10/15/02 **Candle**

## CICS Work Units in Goal Mode

- Address spaces
  - every CICS address space **must** be classified and managed in its native subsystem—JES2, JES3, or STC
  - ▲ **only** velocity goals are supported
  - ▲ the velocity goal for an address space is **ignored** while it is a server
  - Classification and management are **mandatory**—default (ugly) service class is used if not explicitly classified
- Transactions
  - transactions **may** be classified in the CICS subsystem
  - ▲ If in a CICS rule set **any** transaction goes to a service class, a default service class **must** be specified so that **all** transactions are classified
  - ▲ report classes may be used freely—no default is required
  - service class **must** be single period with response time goal
  - ▲ percentile goal matches real world better than average RT
  - ▲ CP/SM reverts to “shortest queue” if percentile goal—requirement?

Page 3

NYCMG 10/15/02



I see many queries based on a misunderstanding of CICS work units and classification restrictions. Let's consider some WLM differences between address spaces and transactions:

<u>Attribute</u>	<u>Address space</u>	<u>Transaction</u>
Classification needed	Mandatory	Optional
Subsystem in which classified	STC or JES	CICS
Default SC	Optional in subsystem	Optional if no other SC
Report classes	Optional	Optional
Consequences of omission	Ugly default SC	No server management
Goal types permitted	velocity	1-period response time (avg. or percentile)
Resource Group	allowed	ignored
Owns resources?	yes	no
Response time reporting?	no	yes
Resource use reporting?	yes	no

Another frequent area of confusion is that “subsystem instances” apply to originating address spaces only (usually TORs) and use the VTAM appl-id in the rule.

Many of the difficulties may be overcome through the judicious exclusion of selected server address spaces from server management.

## How Goal Mode Works (for CICS)

- CICS (as of Version 4) communicates with Workload Manager
  - a Performance Block (PB) is created for each count in MAXTASK
  - CICS puts information in PBs, WLM scans **all of them** each ¼ second\*
    - \* scanning is every 2.5 seconds if region is not managed as a server
  - transaction begins as a new Unit of Work in TOR or single space
    - ▲ CICS issues a classification CALL to WLM → service class
  - each address space transition is reflected in the PBs
- WLM builds topology map of CICS server address spaces for each CICS transaction service class, creates virtual service class per relationship
- CICS and WLM post completed transactions' response times

Page 4

NYCMG 10/15/02



The adjustments made by WLM (SRM) affect only address spaces and enclaves. The intent of adjusting the dispatching priority, I/O priority, and storage protection for server address spaces (as for CICS) is to influence the performance of the served service classes so that they meet their goals.

A server address space is classified in the subsystem in which it starts, to a service class with a velocity goal. CICS transactions are classified in the CICS subsystem, to service classes with response time goals. Understanding this distinction is very important to success with CICS in goal mode.

When WLM goes through its analysis cycle it creates virtual (“internal”) service classes for the interactions of transaction service classes with their respective server address spaces. Evaluation is done for the transaction service classes but the resources live, and are doled out, in the address spaces. The velocity goal of a CICS server space is ignored once it takes on the role of server.

## How Goal Mode Works (for CICS) (continued)

- WLM assesses each CICS service class against its goal and computes a Performance Index every 10 seconds
  - a service class not meeting goal causes a search for resources
  - resources are CPU and I/O priority and dynamic storage protection for each server address space (immediate waste reduction)
  - velocity goal of server address space is ignored
- The degree of success depends on the degree of correspondence between service classes and server address spaces

Page 5

NYCMG 10/15/02

Candle

The key “CICS problem” with goal mode is that the relationship between transaction service classes and server address spaces is ill-defined. The association of transactions with address spaces is a consequence of application implementation decisions by those who were remote in time and knowledge from those implementing goal mode. If the transactions within an address space are homogeneous in execution profile, response time, and importance, they should be in the same service class. If there is a one-to-one correspondence between transaction service classes and server address spaces (AORs) then the performance of the transactions in a service class will be very well controlled by WLM. To the extent that that is not the case, less important transactions will get a free ride and tie up resources that more important work outside of CICS should be able to use.

## Server Management in CICS and IMS

- There are two approaches to managing CICS or IMS “regions”:
  1. set region velocity goals and ignore transaction performance
    - ▲ this is *not* server management—regions are not seen as servers
    - ▲ velocity goals may be inferior to fixed DP and storage isolation
    - ▲ but...velocity goals may be right for test regions
    - ▲ they may also be right for selected high performance regions
  2. set transaction goals and ignore region goals (which must still be set)
    - ▲ region will be on its velocity goal until it becomes a server
    - ▲ region resources will be managed so that transaction goals are met
    - ▲ this *is* server management
    - ▲ most CICS workloads do best with this form of management
    - ▲ region reverts to velocity goal after ~20 minutes of inactivity

Page 6

NYCMG 10/15/02

Candle

Region management is simple. Just assign velocity goals to the CICS regions and let them run that way forever. It is chosen when there is no set of classification rules for the CICS subsystem, or if such rules exist but no transaction is assigned to a service class as opposed to a report class. It is also possible that rules do exist but all the regions are explicitly classified as being subject to region management.

In current systems in which the choice is available, selected regions may be run with velocity goals while others are managed based on transaction goals. There are two extreme cases for choosing velocity goals:

- The work in the region is so important that it has already received special treatment. Its transactions are carefully selected, are likely to be critically important to business success, and have relatively short and homogeneous response times. In this case operating with region goals is simpler and has less overhead than using transaction goals. If a sufficiently high velocity is chosen, at a high level of importance, and if the service class serving the region[s] is also denoted as “CPU Critical”, the Loved One will be treated very well indeed, at some potential cost in holding resources that are not strictly needed all the time.

The region[s] may also be denoted as “Storage Critical” to ensure that pages are stolen at a very low rate in times of inactivity. This extra bit of protection preserves working set across idle intervals to help achieve consistent response times when transactions resume arriving. The “Storage Critical” designation should be reserved for exceptional cases only, for instance when there is danger to life or a possibility of losing real money if a transaction arriving after a period of inactivity is delayed. The image of an Automated Teller Machine in a bad neighborhood in the dark of night comes to mind. “Storage Critical” can be very wasteful if it is too widely applied.

- The work in the region is so unimportant that response time goals can and should be ignored. This is the classical test region or subsystem scenario. In this case, of course, the importance and the velocity goal will both be low and there will be no special protection.

## Server Management in CICS and IMS

- A transaction service class affects regions more or less depending on:
  - whether it meets its goal—if so, not at all
  - residency—if it uses the region more of the time it will have more effect
  - importance—if goals are not met, the most important class is helped
  - degree of missing goal—at equal importance, the most pain is helped
- Therefore:
  - give more important classes difficult but not impossible goals
  - give less important work low importance and very easy goals
- Getting to server management has been regarded as difficult
  - confusion about identifying transactions for classification
  - impression that transactions must be rearranged across regions

## Server Management in CICS and IMS

- It's not that hard!
  - start with a default service class and report classes
  - move rapidly to a more important service class for the *Loved Ones*
  - add more service classes as needed
  - key considerations: which goals are important to meet?
  - OS/390 R10 changes allow special case "opt-out" (e.g. test regions)
  - z/OS 1.2 makes it almost foolproof
- Example:
  - CICS default: 70% complete in 1.0 seconds, importance 4
  - most-loved: 85% complete in 0.6 seconds, importance 2
  - loved: 80% complete in 0.8 seconds, importance 3
  - unloved: 10% complete in 10 hours, importance 5

(importance 1 is reserved for emergency or recovery work)

## Ideal Basis for Distributing Transactions

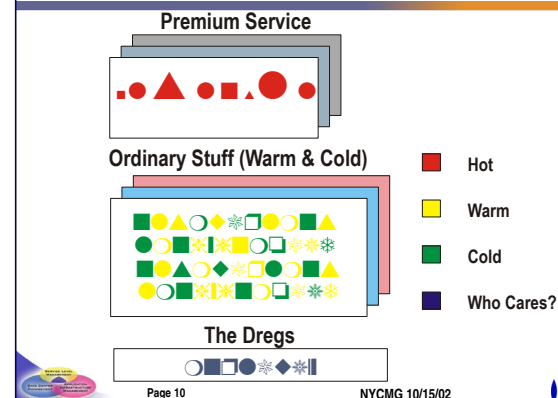
- Isolate mission-critical transactions irrespective of application
  - high priority and adequate storage protection are most effective
  - there are obstacles: affinities, region layout
- Isolate low-importance resource-intensive transactions
  - low priority and no storage protection are acceptable
- Manage the rest of the transactions as before
  - lower priority may be adequate
  - little or no storage protection may yield acceptable performance
  - consolidate underutilized address spaces across applications
- However, identifying the transactions to move may be very difficult in compatibility mode
  - requires monitoring of individual transaction response times
  - requires evaluation against target response times
  - requires freedom to move transactions across applications

Page 9

NYCMG 10/15/02

Candle

## Suggested Arrangement



Page 10

NYCMG 10/15/02

Candle

The ideal is easy to infer from the available information. Assuming we had the freedom to move transactions from region to region, we'd put the most important, response-critical transactions from all applications in one or more dedicated regions, so that the need for high priority and storage protection could be focused on just those transactions that merit such coddling. We might also consider moving the very unimportant transactions that are significant resource consumers to one or more regions of their own, so that they might execute at low priority with no protection.

The vast majority of transactions could be left as is, but the now more homogeneous regions could run at lower priorities than before, with less (or no) protection.

Once the transactions had been rearranged, the original regions could be consolidated and reduced, again combining transactions across applications.

It sounds so easy but it isn't. There are two big problems. The first is securing the freedom to rearrange transactions across regions. To do this requires identifying and eliminating affinities and other interdependencies across transactions in each region. There is also the little matter of identifying the transactions that might be candidates to be moved. A great deal of detailed transaction performance data is needed; it is very difficult to obtain in compatibility mode, and even more difficult to analyze because of the sheer number of objects to consider.

Goal mode changes the picture. All transactions (there may be thousands) must be classified into a small number of service classes. It is far easier to measure and track the performance of a few service classes than of a multitude of transactions.

# Dream On!

## Evolution of CICS-WLM Advice (Lessons in Humility)

- 1995: "It's best to align transaction service classes with server address spaces. Make it so."
  - knocking head against wall
- 1997: "Server management works best and resource waste is minimized when transaction service classes are aligned with server address spaces. Try it. ... Please."
  - begging and pleading
- 1999: "OK, go ahead and stick with velocity goals."
  - giving up
- 2000: Lightning strikes!
  - shows what can happen when you think about it
- 2002: IBM provides a simpler alternative
  - no excuses any more

Unfortunately, goal mode does not change the picture enough. The mere availability of service classes does not serve to get the transactions into them, much less rearrange the transactions across regions.

It was easy in 1995 when nobody was in goal mode to suggest that doing it right had a prerequisite of rearranging the CICS transactions. Nobody paid attention. People started migrating to goal mode, but without defining CICS transaction service classes or classifying transactions.

A couple of years later I had created more and more justification and a methodology that could get CICS into the desired configuration with minimal effort. Again, it didn't work.

Time marched on. Still, very few were in goal mode with CICS transaction goals and virtually nobody went to the effort of moving transactions to different regions. The de facto standard was to run the CICS regions with velocity goals and ignore the transactions.

Finally, in 2000, I had a flash of insight. That was the end of the story until recently

However, in 2002, IBM described a series of enhancements to report classes that had the side effect of creating an absolutely foolproof way of getting to response time goals. Read on....

## What is a Well-Tuned CICS System?

(SRM compatibility mode context)

- Test work is in test regions
- Production work is in production regions
- Intermediate work types may be defined as well
- Each class of region is in a separate performance group
- Transaction performance of *Loved Ones* is measured and monitored
  - DP of production regions is set so that performance of *Loved Ones* is acceptable but not excessive
  - storage isolation is set to protect low-activity *Loved Ones*
- CICS thus has the resources it needs but no more
- Unimportant work is managed appropriately

Page 13

NYCMG 10/15/02

Candle

To understand the new approach, it's important to see what is regarded as "good enough" today. If compatibility mode isn't good enough for CICS, it's a problem with few symptoms.

The essence of a well-tuned CICS system in compatibility mode is the appropriate application of resources to the CICS address space. It is guided by the performance of the most critical transactions. The results achieved are:

- each kind of CICS work (production, pilot, test,...) is in a separate performance group
- for production work, the dispatching priority and storage isolation of its performance group are set so that the key transactions achieve their response time targets consistently
- for other classes of CICS work, sufficient resources go to the address spaces so they achieve adequate performance without delaying other workloads

In short, CICS has the resources it needs and no more.

13

## How Does That Translate to Goal Mode?

- It sure isn't velocity goals!
  - refinement based on transaction performance doesn't work
  - performance may be too good or not good enough if goals are met
  - velocity tends to be set for the worst case
  - at other times, it may be too high
  - the worst part of velocity goals: tuning effort is wasted effort
- It doesn't require rearrangement of transactions across address spaces
  - instead, the same tuning as in optimum compatibility mode is needed

Page 14

NYCMG 10/15/02

Candle

To take CICS from success in compatibility mode to a first success in goal mode, we need to consider what actions in goal mode mirror the best practice in compatibility mode. Let us first recognize that there is no goal mode specification equivalent to a fixed dispatching priority and either fixed or floating storage isolation. The closest we can get is a velocity goal at an appropriate level of importance. However, it's not the same. The priority and protection of a given velocity can vary all over the map depending on workload behavior and contention from other workloads. At some times the DP may be higher than the compat mode fixed priority, and it may be lower at other times.

For initial goal mode success, it's also not necessary to move transactions across regions. They were happy where they were in compat mode; just leave them there until success is solid.

14

## Simple Steps to Server Management

How do we do that? (simplified version—see page 27 for details)

1. set MAXTASK to realistic safe values in all regions
  - ▲ see p. 17 for a better approach as of z/OS 1.2
2. start with velocity goals and a default report class only if you must
  - ▲ e.g. CICSDFLT, 75% complete within 0.5 second, importance 2
  - ▲ importance reflects overall CICS
4. identify the most important and demanding transactions
5. collect response time data in report classes for a while
6. select *Loved Ones*, create a new service class, adjust the default service class and add classification rule[s] as needed
  - ▲ CICSDFLT drops to importance 3
  - ▲ CICSLOVD, 85% complete within 0.6 sec, importance 2
7. split out additional service classes as the need becomes evident

Page 15

NYCMG 10/15/02



So, how to start? I recommend setting the velocity goals and importance levels of the CICS regions for adequate performance of startup and shutdown. Add the CICS subsystem to the classification rules and specify a default report class to gather response time data. You can then add a default service class with a goal representative of good response times. If the typically experienced good response time is a half second, a goal of 0.5 seconds at the 75th percentile will do. Set the importance of the CICS service class to match the business importance of the most important transactions. Most likely, an importance of 2 is a good starting point for a production environment. To avoid excess overhead, make sure the MAXTASK parameter in CICS is only slightly above what is needed.

You have now achieved goal mode with server management, matching and even surpassing the best practice in compatibility mode. Now build on the success by adding report classes and then service classes so that the influence of CICS transactions on server resource management is responsive to performance of transactions within levels of importance. When you add a service class for the loved ones, make sure to adjust the importance of the default service class to reflect the absence of the most important transactions.

## Advantages of the New Approach

- Easy transition to goal mode
- Quick progression to server management
- Step 3 is equivalent to best of compatibility mode
- Steps 6 and 7 improve on that
- Flow of resources to server address spaces is based on current need to meet response time goals
  - velocity will usually be quite low
  - may increase at crunch times
  - resources are recovered
- Involvement of CICS staff is minimal, incremental, and based on their perception of benefit
- Post-R10 changes may be used to help with special needs
  - opt out of server management for test regions
  - add CPU protection for super-hot regions
  - add storage protection for guaranteed 24-hour service situations

Page 16

NYCMG 10/15/02



This approach can get CICS to run in goal mode as well as it did in compatibility mode using WLM's server management capabilities. The only input required from the CICS staff is information on which address spaces are important and which are not. (In a pinch, you might be able to infer this from performance group assignments.)

Before committing to goal mode with server management, make sure that the CICS MAXTASK setting for each region is as low as it can be (just above the high water mark of concurrent active transactions) with a slight margin for safety. Excessive MAXTASKs can lead to excessive WLM overhead.

After you have demonstrated a painless transition to goal mode, check with the CICS wizards to see if they want to get on the bandwagon. They might look at adding service classes to increase the influence of Loved Ones and diminish that of unimportant work. They have to supply the transaction name lists and you can do the rest easily.

With the post-V2R10 changes, you can do a lot more when differences are important: remove selected regions from server management, designate regions or transactions as Storage Critical and service classes as CPU Critical. You can also use much more straightforward classification rules when work is segregated by system.

## Another New Option

- In z/OS Version 1, Release 2, there are changes for report classes:
  - multi-period reporting (no benefit to CICS)
  - response time distribution buckets (BIG benefit to CICS)
- We can now create risk-free migration to response time goals:
  - define all CICS address spaces as “manage to goals of region”
  - create a service class with a goal approximating CICS experience, e.g. CICSRT, 1 second at 85th percentile, importance 2
  - make that service class the CICS default
  - create a report class and add that to the CICS default as on page 21
  - the report class will now produce a set of response time distribution buckets based on the goal response time
    - ▲ note that CICS is still managed to region goals but you now have an accurate estimate of response times
  - use the response time data to create a real service class
  - redefine production regions as managed to transaction goals
- You're done! Now, go back to step 4 on page 15.

## Classifying CICS Address Spaces

```

Subsystem Type . . . . . : STC
Description . . . . . : All started tasks
-----Qualifier-----
Type Name Start Service Report Storage Manage Regions
Critical to Goals of
          DEFAULTS: DISC
          SYSTEM MASTER
1 TN %MASTER%
... ..
1 SY HOTPROD STCMED GENSTC NO
2 TNG HOTCICS VEL60 CICSREGS NO TRANSACTION
2 TN CICSPO3 TOPCICS CICSREGS YES REGION
1 TNG HOTCICS CICS CICSREGS NO TRANSACTION
1 TN CICST* TESTCICS CICSREGS NO REGION
1 TN %%%IRLM SYSSTC IRLMS YES
1 TNG DB2 VEL60 DB2S NO
    
```

**RED** denotes function added in OS/390 V2R10

Here are examples of the new options for classifying address spaces. For CICS and IMS, the “Manage Regions to Goals of” column may be used to choose how resource flow to each address space is managed. The default is “TRANSACTION”, the current method. In this mode, as soon as an address space is recognized to be a server to transaction service classes, its velocity goal is ignored and the regions are managed in the interest of the transaction service classes they serve.

If the “REGION” choice is made, server management is not used. If only some AORs of a CICS subsystem are to be managed with velocity goals, the TOR must not be designated as managed to REGION goal. (Classification of transactions will be bypassed.)

The “Storage Critical” column allows a choice of how the working set of an idle address space will be managed. The choice of “NO” preserves the current aggressive page stealing; “YES” says that the address space’s frame count will stay close to its established high water mark without stealing during periods of inactivity. The “Storage Critical” designation may also be used for transactions classified in the CICS subsystem. Doing so causes the address spaces in which those transactions run to be managed as “storage critical”—if and only if they are managed to transaction goals. Even though “Storage Critical” may be specified for a single rule affecting as little as a single transaction, all transactions of the service class are so regarded and thereby any region in which the service class is active. Stealing is limited to 200 pages per day except when a severe frame shortage exists.

In this example, regions on system HOTPROD get special treatment. CICS regions named in TNG HOTCICS are designated as Storage Critical and are always managed according to the velocity goal of service class VEL60.

HOTCICS regions on other systems receive no special treatment.

Regions whose names begin with the string CICST receive a different kind of special treatment. In this case they are always managed to the velocity goal in service class TESTCICS—presumably a low velocity and importance. This is a way to eliminate the overhead of server management and WLM interaction for test regions.

## CICS Address Space Service Class

```

Service Class Name . . . . . : TOPCICS
Description . . . . . : Super-hot CICS
Workload Name . . . . . : CICS      (name or ?)
Base Resource Group . . . . . :      (name or ?)
CPU Critical . . . . . : YES       (YES or NO)
    
```

```

---Period--- -----Goal-----
# Duration Imp. Description
1          1 Velocity=60%
    
```

## CICS Classification—First Steps (Report Class Only)

```

Subsystem Type . . . . . : CICS
Description . . . . . : IBM-defined subsystem
    
```

```

-----Qualifier----- -----Class----- Storage
Type Name Start Service Report Critical
                        DEFAULTS: _____ ALL_CICS
    
```

Here is the TOPCICS service class. Not only is it Importance 1 with a high velocity goal, it is also designated as CPU Critical, thus giving it a guaranteed high dispatching priority, above those of other work units at lower levels of importance. In this case with Importance 1, only started tasks classified as SYSTEM or SYSSTC will have higher priorities.

## CICS Classification— Single Service Class

Subsystem Type . . . . : CICS  
 Description . . . . : IBM-defined subsystem

-----Qualifier-----			-----Class-----		Storage Critical
Type	Name	Start	Service	Report	
			DEFAULTS:DFLTCICS	ALL_CICS	



## CICS Classification— More Report Classes

Subsystem Type . . . . : CICS  
 Description . . . . : IBM-defined subsystem

-----Qualifier-----			-----Class-----		Storage Critical
Type	Name	Start	Service	Report	
			DEFAULTS:DFLTCICS	NOT_HOT	
1	SI			TEST	
2	TNG			OPFNS	
1	TNG			HOT	
1	TNG			HOT	
1	TNG			NOT_HOT	
1	TNG			OPFNS	



## CICS Classification—Add Loved Ones

Subsystem Type . . . . : CICS  
 Description . . . . . : IBM-defined subsystem

-----Qualifier-----			-----Class-----		Storage
Type	Name	Start	Service	Report	Critical
			DEFAULTS:DFLTCICS	NOT_HOT	
1	SI	CICST*		TEST	
1	TNG	RHLOVED	CICSRHOT	HOT	
1	TNG	LOVED		HOT	
1	TNG	UNLOVED		NOT_HOT	
1	TNG	OPFUNCS		OPFNS	



## CICS Classification— Add *Really* Loved Ones

Subsystem Type . . . . : CICS  
 Description . . . . . : IBM-defined subsystem

-----Qualifier-----			-----Class-----		Storage
Type	Name	Start	Service	Report	Critical
			DEFAULTS:DFLTCICS	NOT_HOT	
1	SI	CICST*		TEST	
2	TNG	OPFUNCS		OPFNS	
1	TNG	RHLOVED	CICSRHOT	HOT	
2	UI	TOPDOG	SUPERHOT	HOT	YES
1	TNG	LOVED		HOT	
1	TNG	UNLOVED		NOT_HOT	
1	TNG	OPFUNCS		OPFNS	



## CICS Classification— More Service Classes

Subsystem Type . . . . . : CICS  
 Description . . . . . : IBM-defined subsystem

-----Qualifier-----			-----Class-----		Storage
Type	Name	Start	Service	Report	Critical
			DEFAULTS:DFLTCICS	NOT_HOT	
1	SI	CICST*	CICSLOW	TEST	
2	TNG	OPFUNCS	CICSLOW	OPFNS	
1	TNG	RHLOVED	CICSRHOT	HOT	
2	UI	TOPDOG	SUPERHOT	HOT	YES
1	TNG	LOVED	CICSHOT	HOT	
1	TNG	UNLOVED	CICSLOW	NOT_HOT	
1	TNG	OPFUNCS	DFLTCICS	OPFNS	

## Goal Definition for a CICS Service Class

Service Class Name . . . . . : TOPTRAN  
 Description . . . . . : Time-critical CICS  
 Workload Name . . . . . : CICS (name or ?)  
 Base Resource Group . . . . . : (name or ?)  
 CPU Critical . . . . . : YES

---Period---		-----Goal-----	
#	Duration	Imp.	Description
1	2	85%	complete within 00:00:0.500

Here is a service class definition for a CICS transaction response time service class. Note that it has only one period. Since transaction response time service classes do not own resources, they do not accumulate service and thus cannot move through periods. Note also that it is a percentile goal.

This service class is not in a resource group, and it is designated as CPU Critical. Every address space in which it runs “inherits” this attribute and level of importance until the service class ceases to be active for approximately 20 minutes.

If CP/SM is used to determine transaction placement based on WLM data, the goal must be an average RT goal, as of OS/390 V2R10. The value of percentile response time may outweigh the slight loss of CP/SM effectiveness when it reverts to the “shortest queue” method of determining transaction routing.

## Details of Classifying CICS Transactions

- A single classification tree ignores reality:
  - not all CICS regions are equally important
  - not all important CICS regions are alike
  - same transactions may occur in all kinds of regions
- Therefore:
  - you might need more than one default service class
    - ▲ the main default should be good for the most transactions
    - ▲ other defaults apply per set of address spaces
  - first-level rules should carve out regions (TORs) as SI or SIG rules
    - ▲ service class on SI[G] rule is for those regions' default
  - lower-level rules are usually based on transaction names (TNG)
    - ▲ aggressive goals and high importance for traditional loved ones
    - ▲ relaxed goal and moderate importance for less-loved
    - ▲ now comes the default
    - ▲ truly unloved or long-running with low importance on the bottom
  - ◆ ensure that this goal will always be met

Page 27

NYCMG 10/15/02

Candle

In the real world, there are usually many CICS regions, differing by business importance and applications served. A single default transaction service class is rarely applicable across all regions. Fortunately the classification capabilities of goal mode allow you to choose different defaults for each address space or group of address spaces.

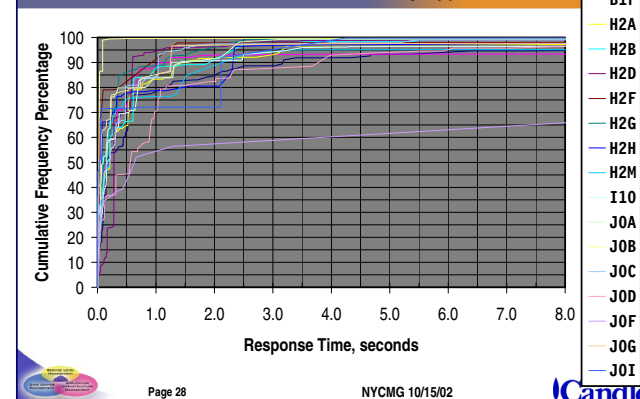
The rules may get a bit cluttered but if the transactions of the most critical regions are classified first, performance will be affected very little.

For each significant production region, three or four service classes may be enough:

- one for top performance
- a second for slightly less critical transactions
- the third for the great majority of transactions—the local default
- finally, a service class with a very easy goal and low importance to ensure that unimportant or conversational transactions have very little influence on the management of resource flow to the region(s).

27

## Some Real Data CFDs of 95th Percentiles by ApplID



Page 28

NYCMG 10/15/02

Candle

Here is data from a customer site, showing 17 of about 35 CICS regions on a single system. These are all independent regions (no MRO). What is plotted is the cumulative frequency of response times at the 95th percentile per transaction name. Note that *most* of the regions would do well if the transaction response time goal is 1 second at the 84th percentile. However, six of the regions cannot achieve this goal and one will do much better. Additional service classes will be needed to allow transactions in these regions to meet their goals and achieve appropriate PIs.

These classes would include:

- 1 second at the 54th percentile, for one region
- 1 second at the 72nd percentile, for two regions
- 1.5 seconds at the 80th percentile, for three regions
- 0.25 seconds at the 95th percentile, for the wild overachiever

We retain the default of 1 second at the 85th percentile at importance 3, as well as the service classes for Loved Ones and The Unloved.

28

## CICS rules in a "Real" Service Definition

Default service class is CICSDFLT (P85S10I3)

Qualifier #	Qualifier type	Qualifier name	Starting position	Service Class
1	TNG	STARS		CICSLOVD (P85S10I2)
1	TNG	WORST		CICSUGLY (P10H2I5)
1	SIG	FASTEST		P90S07I3
2	TNG	EXCEPT1		P80S40I3
1	SIG	SLOWER		P85S15I3
1	SIG	SLOWEST		P80S40I3
1	SIG	WEIRD1		P75S20I3
1	SIG	WEIRD2		P55S30I3

Pointer to group of server regions named as VTAM APPLIDs

Pointer to group of exception transactions

"Default" for this group of regions

Page 29

NYCMG 10/15/02

Candle

Here are the classification rules that apply to this example. CICS transactions that do not match any of the specific rules are assigned to service class CICSDFLT. No report classes are used and no selections are marked as Storage Critical. Response time distribution patterns of high volume transactions and of many CICS address spaces were studied. Since this site had a separate production sysplex, there was no need to distinguish between test and production regions. Two groups of transactions were treated globally across all regions, STARS and WORST. Groups of regions are then classified, from FASTEST to SLOWEST and then two groups with unusual characteristics, WEIRD1 and WEIRD2. In the fastest group of regions, a subset of transactions behaved atypically and are classified in the second level rule to a different service class.

The service classes other than the default and the two global ones are for the purpose of this example systematically named and defined. However, such names are not recommended for a production policy, since the shorthand notation loses a valuable level of indirection and late binding.

For example, P85S15I3 means "Percentile 85, 1.5 seconds, importance 3." Classes CICSDFLT, CICSLOVD, and CICSUGLY would have systematic names of P85S10I3, P85S10I2, and P10H20I5 respectively. That last really does mean "tenth percentile, 2 hours, importance 5." Such a service class is as close as CICS can get to discretionary and really means that such transactions should have no practical effect on the management of the regions.

The example cannot be freely generalized to sites that have a different configuration or workload. A careful study of the workload is necessary before setting up CICS rules and service classes.

## CICS Gotchas

- CICS transaction classification in CICS subsystem is all-or-none (but see R10 changes)
  - use subsystem instance (SI), logical unit (LU) or user-ID (UI) rules to distinguish loved ones from the unloved if the transaction names are the same in multiple TORs or subsystems. As of R10, you can also classify based on sysplex ID.
  - "subsystem instance" is a VTAM applid, applies only to the address space requesting classification, usually a TOR or single CICS region
- Excessive MAXTASKs can increase WLM overhead
- CICS regions are classified in the subsystem(s) in which they run
  - STC is "normal"
  - JES2 or JES3 may be used for various reasons (re-justify?)
  - the only goal type supported is velocity—(ignored as soon as WLM recognizes the address space as a server)
- CPSM doesn't understand percentiles—sounds like a requirement

Page 30

NYCMG 10/15/02

Candle

Here are reminders of some typical problems encountered when running CICS in goal mode.

One problem is the all-or-nothing nature of running with transaction response time goals. However, it need not be a problem if the flexibility of the classification scheme is exploited. Suppose that transaction BUZZ is very important on System A and just one of the players on the rest of the systems. Assuming that the VTAM appl-IDs of the TORs on the systems are different (as they should be), a Subsystem Instance rule can pick off System A's CICS first and classify BUZZ as going to service class CICSBOOT. BUZZ on other systems can be classified differently or just allowed to default to service class ROUTINE.

The post-V2R10 changes make it easier and more direct to differentiate work according to the system on which it executes.

The interface between CICS and WLM depends on the construction, updating, and polling of data fields called Performance Blocks (PBs). Once an address space is recognized as a server, a number of PBs equal to MAXTASKS is built. If MAXTASKS is excessive across a large number of CICS regions, WLM overhead can rise unnecessarily. Cut MAXTASKS to a realistic value.

Another source of confusion is the need to classify two different sets of entities two different ways—CICS address spaces in the subsystems in which they run (STC or JESx) and CICS transactions in the CICS subsystem. The permitted goal types are different, and the use of the goals is different. Once the relationship between server address spaces and the service classes they serve is established, the address space becomes a totally dependent entity, managed on behalf of its client service classes. Transaction service classes do not accumulate service or have measured service rates; therefore they cannot be in resource groups or have multiple periods.

A wrinkle appears when the CICSplex Service Manager (CPSM) is being used to route transactions based on goal attainment: CPSM understands only average response times, not percentiles. The need for a user group requirement is obvious. (The alternative is not to use the WLM option of CPSM if percentile response time goals are to be used.)

One last question: can anybody explain why CICS should be run as a batch job?

## Potential Problems and Remedies

- If you have many CICS transactions not originating from a CICS terminal, WLM may not see them and won't classify them
  - It's not WLM's fault—classification must be requested by CICS for each way a transaction is launched
  - the application must name the transaction to be used; many don't
  - result: WLM sees the name of a service transaction, not the real one
    - ▲ MQ-initiated transactions (possibly implementation choice)
    - ▲ “umbrella” transactions in application packages
    - ▲ other spun-off transactions such as mirror transactions
    - ▲ transactions arriving from CICS Transaction Gateway
- Another bad habit: all-purpose transactions with variable response times
- Redistributing transactions in CICS is difficult but worth the effort
  - ownership politicizes the problem and complicates solutions
  - affinities are the main technical problem
  - shared tables and VSAM Record Level Sharing are mechanisms that can clear the way to ideal transaction placement

Page 31

NYCMG 10/15/02



The interactions between CICS and WLM are complex. Part of the interface is the explicit invocation of WLM classification by CICS as each transaction is launched. However, there are numerous ways in which CICS transactions are launched, and some of them have not included the call to classification. There may even be application design considerations that cause classification to be bypassed.

The practical effect has been that at least some MQ-initiated transactions, some transactions created by Umbrella Transaction Services, and some other transactions originated within CICS, are not classified and therefore do not contribute to the service classes that influence the operation of server address spaces. CICS Transaction Server closes some of these gaps in its newest release.

According to a principal CICS developer, the problem is that the application need not fill in a transaction name if it invokes a processing program by name. The CICS-WLM interface does not deal in program names, only transaction names, so that WLM falls back on the service transaction name rather than the “real” transaction name. If the application is home grown, the problem of imprecise classification may be easily corrected. A solution may be difficult to achieve if the application is vendor-supplied.

Some “dead hand of the past” issues limit an installation's ability to move transactions freely across address spaces. Address space affinities head the list. Fortunately, there are opportunities to dispose of affinities by using newer structures and services within the CICS subsystem. The investment will be worth it.

## Summary

- Revolutionary change is not going to happen
- CICS wizards may need to be dragged along kicking and screaming—
  - or be made to see a self-interest in making changes
- Make incremental changes with obvious benefit at each step of the way
  - it may take years!
- Start with a default service class for all CICS transactions
  - consider server management opt-out (R10+) for test regions
  - if nervous, start with velocity management only for a limited time
- Add report classes to get response time data
- Subdivide service classes according to transaction performance expectations and importance
  - change effect of each service class on server resource management
  - refine classification rules accordingly
- Move extreme transactions to separate address spaces
  - reduce resource waste and isolate premium service classes
  - consider CPU and Storage Critical for only critical regions or classes

Page 32

NYCMG 10/15/02



Once you get over the idea that the CICS staff will drop everything and rush to do your bidding just because you ask, a more practical approach to CICS exploitation of goal mode may be considered.

The usual first step is to avoid the hard questions entirely and set velocity goals for the CICS address spaces—and then walk away. Doing that, however, perpetuates the compatibility mode situation and may even make it worse. At times of constraint, achieving the set velocity may require more dispatching priority and/or storage protection than was necessary in compatibility mode. Better to start with a default CICS service class.

You have now achieved server management without any knowledge of the transaction set. Now the velocity is a consequence of how you are managing CICS, not the means of management.

To go on, add report classes for specific lists of transactions. Again, after analyzing data, add service classes so that the more important transactions tend to be the ones that determine server resource distribution.

After it becomes obvious that some transactions deserve special treatment, put the most-loved ones in separate address spaces, and the least-loved in other separate address spaces.

This whole evolution may take several months, but each month you'll be wasting less resources and managing CICS better—and the CICS wizards can become your cooperating allies.